Ontology Integration by Propagating a Context in Priorly Matchable Concepts

by
Duong Trong Hai

A THESIS
Submitted to the faculty of
INHA UNIVERSITY
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

School of Computer Science and Information Engineering
February 2012
This manuscript has been read and accepted for the Graduation Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy

Professor Kang, Sanggil, Ph.D

Professor Jo, Geun Sik, Ph.D (Supervisor)

Professor Lim, Joon Sik, Ph.D

Professor Lee, Sang-Chul, Ph.D

Professor Lee, Jae-Dong, Ph.D

School of Computer Science and Information Engineering
Inha University
February 2012
Abstract

The main aim of this research is to deal with wide-scale semantic heterogeneity in ontology matching. Although several efforts in ontology matching have already been contributed, they have different focuses, assumptions, and limitations. A common point among existing methods involve blind or exhaustive computing similarities among all entities (concepts and properties) across different ontologies that leads to mismatching problem. The main focus for this research is to avoid computing similarities between mismatching concepts and unmatchable concepts across ontologies for enhancing matching accuracy and guaranteeing high performance. To accomplish this, instead of computing similarities between all combinations of concepts across the ontologies, matchable concepts would be the best to propagated in a priority policy during matching process.

Therefore, the Priorly Matchable Concepts (PMC) propagation method is proposed. The priorly matchable concepts are determined by two measurements, Concept Types and Concept Importance. All concepts belonging to an ontology are classified into disjoint categories by analysing taxonomy dependency and types of properties. The innovation behind the Concept Types is to supply an additional suggestion for identifying possible matching concepts, regarding that if two concepts are semantically equivalent, then they must be classified within the same concept type. The Concept Importance, in other hand, is a measurement of the importance of a concept that shows how centrally located and richly described a concept is in the taxonomy of an ontology. It means that the measure must take into account contributions from the concept’s attributes
and from all other concepts in the ontology through their relations. The 
*Concept Importance* is to supply an identifiable link between two hetero-
genous descriptions of a concept, regarding that if two ontologies need 
to be integrated, two concepts in different ontologies should be priorly 
computed for similarity if their importance-score are most similar than 
other pairs. A novel idea of PMC propagation method is that direct 
concept matching is driven from *Concept Types* and *Concept Importance*. 
Consequently, PMC propagation method supports content-based matching in a less complexity and name-base matching avoiding mismatching concepts. An application of PMC propagation method for developing an effective matching algorithm (called *Anchor-Prior*) are proposed. The key idea of the called *Anchor-Prior* algorithm is to start from an *Anchor* (two matched concepts) to work towards a collection of matched pairs among its neighboring concepts by computing similarities between the priorly collected concepts across the ontologies starting from the anchor.

The method is implemented in Java for matching between OWL ontologies by utilizing Jena OWL API. Moreover, an experiment is done in Benchmark data set of OAEI (Ontology Alignment Evaluation Initiative)\(^1\) and Linked open data (LOD) set. The effectiveness of *Anchor-Prior* is shown in terms of precision and recall in comparison with the best representative methods for Benchmark data set 2009 \(^2\) and LOD set. 

**Keywords:** Ontology, Ontology Integration, Ontology Matching, Ontol-
ogy Alignment, Ontology Merging, Conflicts, Inconsistency.

---

\(^1\)http://oaei.ontologymatching.org
\(^2\)http://oaei.ontologymatching.org/2009/benchmarks/
Contents

Abstract i

List of Figures vii

List of Tables ix

1 Introduction 1
  1.1 Motivation ........................................ 2
  1.2 Problem Definition ............................... 8
      1.2.1 Ontology Matching .......................... 8
      1.2.2 Ontology Alignment ......................... 9
      1.2.3 Ontology Mapping ............................ 10
      1.2.4 Ontology Merging ............................ 10
  1.3 Goals of the Dissertation .................... 11
  1.4 Overview of Priorly Matchable Concepts Approach .... 12
  1.5 Related Work .................................. 14
      1.5.1 Lily ....................................... 15
      1.5.2 RiMOM ..................................... 16
      1.5.3 ASMOV .................................... 18
      1.5.4 Anchor-PROMPT .............................. 20
      1.5.5 Anchor-Flood ................................ 21
  1.6 Thesis Outline ................................. 23

2 Ontology Engineering 25
  2.1 Basic Ontology Concepts ..................... 25
      2.1.1 What is an Ontology? ........................ 25
      2.1.2 The Main Components of an Ontology ...... 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.3</td>
<td>Types of Ontologies and their Uses</td>
<td>28</td>
</tr>
<tr>
<td>2.1.4</td>
<td>Ontology Model</td>
<td>30</td>
</tr>
<tr>
<td>2.2</td>
<td>Methodologies for Ontology Building</td>
<td>32</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Criteria for Designing Ontology</td>
<td>32</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Engineering-oriented Method</td>
<td>33</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Collaborative-oriented Method</td>
<td>34</td>
</tr>
<tr>
<td>2.3</td>
<td>Ontology Representation Languages and Development</td>
<td></td>
</tr>
<tr>
<td>2.3.1</td>
<td>Ontology Representation Languages</td>
<td>41</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Ontology Development Tools</td>
<td>44</td>
</tr>
<tr>
<td>3</td>
<td>Similarity Techniques for Ontology Integration</td>
<td>47</td>
</tr>
<tr>
<td>3.1</td>
<td>Basic Techniques</td>
<td>47</td>
</tr>
<tr>
<td>3.1.1</td>
<td>String-based Similarities</td>
<td>48</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Language-based Similarity</td>
<td>50</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Linguistic-based Similarity</td>
<td>51</td>
</tr>
<tr>
<td>3.2</td>
<td>WordNet-based Similarity</td>
<td>52</td>
</tr>
<tr>
<td>3.3</td>
<td>Lexico-Syntactic Patterns-based Similarity</td>
<td>54</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Lexico-Syntactic Patterns for Hypernym</td>
<td>54</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Noun Phrase Analysis</td>
<td>55</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Lexico-Syntactic Patterns for Similarity Measure</td>
<td>56</td>
</tr>
<tr>
<td>3.4</td>
<td>Collaborative Acquisition Algorithm</td>
<td>59</td>
</tr>
<tr>
<td>3.5</td>
<td>Ontological Similarities</td>
<td>60</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Name Similarity</td>
<td>60</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Annotation Similarity</td>
<td>60</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Structure Similarity</td>
<td>61</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Aggregated Similarity</td>
<td>62</td>
</tr>
</tbody>
</table>
# 4 Ontology Integration by Priorly Matchable Concepts Propagation

## 4.1 Priorly Matchable Concepts Notion

## 4.2 Priorly Matchable Concepts Generation

### 4.2.1 Concept Types

### 4.2.2 Concept Importance

### 4.2.3 PMC Mapping Functions

## 4.3 PMC Propagation-based Matching Algorithm

### 4.3.1 Basic Notion

### 4.3.2 Anchor-Prior Algorithm

# 5 Experimental Result

## 5.1 Data Sets

## 5.2 Evaluation Method

## 5.3 Evaluation Result

### 5.3.1 Similarity Techniques Evaluation

### 5.3.2 Evaluation Based on Benchmark Track

### 5.3.3 Evaluation Based on LOD Schema

## 5.4 Discussion

### 5.4.1 Concept Types

### 5.4.2 Concept Importance

# 6 Conclusion

## 6.1 Summary of Contributions

## 6.2 Advantages and Limitations

## 6.3 A Brief History of Research Progress

## 6.4 Future Directions
REFERENCES 113

Acknowledgement 128
List of Figures

1.1 Semantic interoperability among distributed information systems ........................................ 5
1.2 Query between information systems ................................................................. 6
1.3 Communication between agents ................................................................. 7
1.4 System architecture of LILY ................................................................. 15
1.5 System architecture of RiMOM ................................................................. 17
1.6 System architecture of ASMOV ................................................................. 19
1.7 System architecture of PROMPT ................................................................. 20
1.8 System architecture of Anchor-Flood ......................................................... 22

2.1 Fragments of ontologies ................................................................. 27

3.1 Language-based similarity process ......................................................... 50
3.2 The acquisition process ................................................................. 59

4.1 A simple algorithm for concept types classification ................................ 70
4.2 Concept types and concept importance ....................................................... 73
4.3 PMC for ontologies #101 and #304 ......................................................... 80
4.4 Initial anchor between ontologies #101 and #304 ....................................... 81
4.5 Segments of neighbors ................................................................. 82

5.1 A comparison of the similarity methods over all Benchmark data set ............ 92
5.2 Experimental result comparison in Recall, and Precision over Benchmark data set ........................................ 94
5.3 Anchor-Prior (Aprior) compares with the best representative methods for the Benchmark data set . . . . . . . . . . . . . . . . 95
5.4 A comparison between the number of computed pairs of concepts and the number of all combination of concepts . . 96
5.5 A comparison between pre-process running time and matching process running time on Benchmark data set . . . . . 97
5.6 Running time comparison between Aprior and FOAM on Benchmark data set . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 98
5.7 LOD schema evaluation comparison between Aprior and the previous works BLOOMS, BLOOMS+, and S-Match . . 100
5.8 LOD schema evaluation comparison between Aprior and BLOOMS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101
5.9 Concepts distribution in the ontological taxonomy . . . . . 103
List of Tables

3.1 The 16 patterns of noun phrase ................. 57

4.1 A classification of concepts belonging to ontologies #101 and #304 ............................ 79

5.1 Sample of correct matches from LOD ontologies to PROTON generated by Aprior .................. 99

5.2 Sample of incorrect matches from LOD ontologies to PROTON generated by Aprior .............. 99
Chapter 1

Introduction

Recently, ontologies have moved from a topic in philosophy (theory of existence) to a topic in applied artificial intelligence that is at the center of modern computer science. Since the beginning of the nineties ontologies have become a popular research topic, investigated by several artificial intelligence research communities, including knowledge engineering, natural language processing and knowledge representation. More recently, the ontology is also becoming a backbone technology for the Semantic Web that is an extended Web of machine-readable information and automated services that extend far beyond current capabilities. Its importance is being recognized in a multiplicity of research fields and application areas, such as database design and intelligent information integration, cooperative information systems, information retrieval and extraction, electronic commerce, government intelligence and knowledge management. For an information system, an ontology is a representation of some preexisting domain of reality which: (1) reflects the properties of the objects within its domain such that there is a systematic correlation between reality and the representation itself, (2) is intelligible to a domain expert, and (3) is formalised in a manner supporting automatic information processing. The ontologies have been developed to provide machine-processable semantics of information sources that can be communicated between different agents (software and humans). Such ontologies play a central role in facilitating data exchange between several sources. To do this efficiently, distributed ontologies have to be in-
tegrated. Ontology integration is consisting of sub-problems as follows: Mapping mechanisms need to bridge their knowledge gaps. Ontology matching (or alignment) is a process of finding correspondences between semantically related entities in heterogeneous ontologies. Ontology merging is a process of generating an ontology from heterogeneous ontologies that best represented them.

This chapter is organised into five sections. First section is dedicated to the motivation of the ontology integration problem. The motivation is begun by showing that information heterogeneity as a big obstacle to achieve semantic interoperability in the WWW and current solutions are far from enough for this problem. Next, ontology integration is as a fundamental component in the Semantic Web vision to solve semantic interoperability problem. In second section, the ontology integration problem is technically defined in various instances of ontology integration occurring in different contexts, such as folksonomies, classifications, databases, XML and entity relationship schemas and finally formal ontologies. Third section, the goals of the dissertation are discussed along with challenge of the ontology integration problem. The fourth section, the proposed solutions for ontology integration are discussed along with the contributions to the ontology integration community. Finally a road map to the rest of the dissertation is given.

1.1 Motivation

Semantic interoperability is the ability of two or more computer systems to exchange information and have the meaning of that information accurately and automatically interpreted by the receiving system. Achieving
semantic interoperability among different information systems is very
laborious, tedious and error-prone in a distributed and heterogeneous
environment like the World Wide Web (WWW).

Mid-90s, Extensible Markup Language (XML) was developed from
GSML (Standard generalized Markup Language (SGML ISO 8879: 1986))
to provide a set of metadata tags expecting representing Semantic Web
data, but XML does not define the meaning of the tags. Thus, available
information on the Web can only be accessed with syntactic interoper-
ability. Software agents can not understand and process information
effectively. In the late 1990s, the World Wide Web Consortium (W3C)
Metadata Activity started work on RDF Schema (RDFS), a language for
RDF vocabulary sharing. The RDF became a W3C Recommendation in
In 2001, Tim Berners-Lee and colleagues have established a vision of the
Semantic Web [4].

In 2005, at the OWL Experiences And Directions Workshop a consen-
sus formed that recent advances in description logic would allow a
more expressive revision to satisfy user requirements more comprehen-
sively whilst retaining good computational properties.

The ontologies is to allow users to organize information on the tax-
onomies of concepts, with their own attributes, to describe relationships
between the concepts. When data is represented by using the ontolo-
gies, software agents can better understand the content of the data and
messages, smarter integrate data for multiple jobs than the conventional
web used. In this way, the sites not only performs the function of re-
ceiving and displaying information but also the ability to automatically
extract information, query, argue the knowledge base to give accurate information automatically.

An important application area for ontology is the integration of existing systems. In order to enable machines to understand each other it is necessary to explicate the context of each system in a formal way. Ontologies are then used as interlingua for providing interoperability since they serve as a common format for data interchange. Such a feature is specially desirable in large scale web commerce environments. The e-business is about trading relationships between commercial entities (business-to-business, or B2B) [85]. The ontology is a structured conceptual model that is used to define semantics for the e-commerce domain. It supports semantic search and navigation using product and service knowledge by prospective buyers to discover what to buy, and subsequently to determine pricing and availability. In this case, the relatively static knowledge of the ontology maps to the relatively dynamic data of the vendors. Furthermore, an ontology can model not only commodities, but also agents, i.e., buyers and sellers, both human and artificial. Using user profile to assist the search process, queries can be customized to a users known functions and interests, possibly based on previous interaction with that user.

Ontologies provide a shared and common understanding of a domain of interest, which may be used as a unifying framework, and can be communicated between people and heterogeneous and distributed application systems, since ontologies formally describe terminological concepts and their relationships in a specific domain of interest. If there is a semantic defined in applications, then this problem can be resolved.
Figure 1.1: Semantic interoperability among distributed information systems

Therefore, ontology-based systems are identified as the solution for a problem of communication between dissimilar software applications, even applications that use different ontological standards of specific domains and can be solved by using transformation mappings\(^1\) (see Figure 1.2).

Agent systems are software programs that possess domain knowledge. In knowledge-level [77] communication ability, the agent is able to work with other agents whether in communication, interaction or any social field that needs multiple agents in a system (MAS). For example, if these agents need each other to complete goals, some agents need to do the task and instructions, give a response or feedback, or exchange information. In this situation, the software agents become intelligent

\(^1\)http://www.cs.man.ac.uk/seanb/teaching/COMP30411/Mapping.pdf
because they can make use of the knowledge contained in an ontology (common structure of knowledge base) to use in process of negotiation and decision-making.

As another example, a D2D exchange, standardized product catalogues could be represented by a well-defined ontology which is collaboratively developed by experts. Therefore, an outsourced translation service can be used, through a company, which enables communication between different departments or business units using different terminologies. As another example, a FOAF is a common understanding of user profile, a well-defined ontology describing relationships among users, their interests and information. This ontology is a global understanding which has been applied by most social webs as its knowledge base. Therefore, the social webs more easily share knowledge.

In an enterprise application integration (EAI) that deals with the problem of data, process, application, and business integration within
Figure 1.3: Communication between agents

One or between several organizations. The ontology aim at providing a shared, common understanding of data, services and processes within enterprise application integration. For example, the Enterprise Interoperability Ontology (ENIO) was proposed in [5], an EAI Ontology that captures and represents formally all entities involved in the EAI scenarios, i.e. data, services and processes. The goals of ENIO is to enhance message exchange between services, to provide a reference model of data semantics, and to enable effective search and discovery of services.

Ontologies have been evolved in knowledge management as a knowledge representation to provide computer systems with a conceptual yet computational model of a particular domain of interest. Ontologies are also applied on the development of concepts, methods, and tools supporting the management of human knowledge. In this way, computer systems can base decisions on reasoning about domain knowledge, similar to humans.
1.2 Problem Definition

In the literature, there are many definitions of ontology integration ([39, 89, 82, 65, 67] as example) but the definition given by [95] is refereed in this research that defined as the process of finding commonalities between two different ontologies $O$ and $O'$ and deriving a new ontology $O^*$ that facilitates interoperability between computer systems that are base on the $O$ and $O'$ ontologies. The new ontology $O^*$ may replace $O$ or $O'$, or it may be used only as an intermediary between a system based on $O$ and system based on $O'$.

Depending on the amount of change necessary to derive $O^*$ from $O$ and $O'$, different levels of integration can be distinguished such as ontology matching/alignment, ontology mapping, and ontology merging.

1.2.1 Ontology Matching

Ontology matching aims at finding correspondences between semantically related entities of different ontologies. These correspondences may stand for equivalence as well as other relations, such as consequence, subsumption, or disjointness, between ontology entities. Ontology entities, in turn, usually denote the named entities of ontologies, such as classes, properties or individuals. However, these entities can also be more complex expressions, such as formulas, concept definitions, queries or term building expressions.

Cupid [65] implements an algorithm comprising linguistic and structural schema matching techniques, and computation of similarity coefficients using domain-specific thesauri.
H-Match [7] is an automated ontology matching system. H-Match inputs two ontologies and outputs (one-to-one or one-to-many) the correspondences between concepts of these ontologies with the same or closest intended meaning. The approach is based on a similarity analysis via affinity metrics, e.g., term-to-term affinity, data type compatibility, and thresholds. H-Match computes two types of affinities (in the [0,1] range), viz., linguistic and contextual. These are then combined via weighting schemas, thus yielding a final measure, viz., semantic affinity. Linguistic affinity builds on the thesaurus-based approach of the Artemis system.

COMA [11] is a schema matching tool based on parallel composition of matchers. It provides an extensible library of matching algorithms, a framework for combining obtained results, and a platform for evaluating the effectiveness of the matchers. Most implement string-based techniques, such as affix, n-gram, edit distance; others share techniques with Cupid, e.g., thesauri look-up.

1.2.2 Ontology Alignment

Ontology alignment is a set of correspondences between two or more (in case of multiple matching) ontologies (by analogy with molecular sequence alignment). The alignment is the output of the matching process. Alignments can be used for various tasks, such as ontology merging, query answering, data translation or for browsing the Semantic Web. In the above mentioned example, the library can take advantage of alignments for automatically ordering a book and the seller can use them for checking the availability of a reference by the library.
1.2.3 Ontology Mapping

Ontology mapping is the oriented, or directed, version of an alignment: it maps the entities of one ontology to at most one entity of another ontology. This complies with the mathematical definition of a mapping instead of that of a general relation. The mathematical definition would in principle require that the mapped object is equal to its image, i.e., that the relation is an equivalence relation. A mapping can be seen as a collection of mapping rules all oriented in the same direction, i.e., from one ontology to the other, and such that the elements of the source ontology appear at most once.

RiMOM [62] (Risk Minimisation based Ontology Mapping) is an approach inspired by Bayesian decision theory, which formalizes ontology matching as a decision making problem. Given two ontologies, it aims for the optimal and automatic discovery of alignments, which can be complex (for example including concatenation operators). The approach first searches for concept-to-concept correspondences, and then searches for property-to-property correspondences.

1.2.4 Ontology Merging

Ontology merging is the creation of a new ontology from two, possibly overlapping, source ontologies. The initial ontologies remain unaltered. The merged ontology is supposed to contain the knowledge of the initial ontologies, e.g., consequences of each ontology are consequences of the merge. This concept is closely related to that of schema integration in databases.

FCA-Merge is a method for merging ontologies, which is a bottom-
up approach supporting a global structural description of the merging process. For the source ontologies, it extracts instances from a given set of domain-specific text documents by applying natural language processing techniques. Based on the extracted instances, mathematical techniques from formal concept analysis are applied. The produced result is explored and transformed to the merged ontology by the ontology engineer.

OntoMerge [16] is a system for ontology translation of the Semantic Web. Ontology translation refers to such tasks as (i) dataset translation, that is, translating a set of facts expressed in one ontology to those in another ontology; (ii) generating ontology extensions, that is, given two ontologies $O$ and $O'$ and an extension (sub-ontology) $O_s$ of the first one, build the corresponding extension $O'_s$, and (iii) query answering from multiple ontologies. The main principle of this approach is ontology translation via ontology merging and automated reasoning.

This dissertation focuses on ontology matching that is the process of finding commonalities between two different ontologies.

1.3 Goals of the Dissertation

The backbone technology for Semantic Web is ontologies. Ontologies provide a shared understanding of certain domains that can be communicated between people and application systems. Ontologies define formal semantics for information, thus they can be used to represent the semantics of structured information enabling sophisticated automatic support for acquiring, maintaining, and accessing information, knowledge sharing and reuse. Therefore, the ultimate goal of this research is
to solve the problem of ontology matching, and thus enable semantic interoperability between different application systems such as web applications and services. More specifically, the aim of this thesis is how to deal with wide-scale semantic heterogeneity in matching between large ontologies. This is because a common point among existing methods involve blind or exhaustive computing similarities among all entities (concepts and properties) across different ontologies. In the case of wide-scale semantic heterogeneity, in order to decide semantic commonalities between concepts, those methods need to analyze the similarities between all related properties and instances, the computational complexity will be rapidly increased. In addition, possible commonalities between two ontologies are determined by the similarity of entity names leading to semantic mismatches and logical inconsistencies. Since the name of a concept cannot express the precise semantics of the concept. In practice, two concepts with the same name may have different semantics, or two differently naming concepts may have the same semantics, and there are many concepts with similar names such as *Book* and *InBook* or *Publisher* *Published* in the same ontology. To accomplish the major aim and focuses, the instead of computing similarities between all combinations of concepts across the ontologies, matchable concepts should be propagated in a priority policy during matching process.

### 1.4 Overview of Priorly Matchable Concepts Approach

In this research, two issues are highlighted. The first issue is how to avoid computing similarities between mismatching concepts across ontologies. The second issue is how to reduce complexity, concerning wide-scale
semantic heterogeneity in content-based matching. To accomplish the major aim and focuses, the instead of computing similarities between all combinations of concepts across the ontologies, priorly matchable concepts are propagated in a priority policy during matching process. The priorly matchable concepts are determined by two measurements Concept Types and Concept Importance. All Concepts belonging to an ontology are classified into disjoint categories by analysing taxonomy dependency and types of properties. The innovation behind Concept Types is to supply an additional suggestion for identifying possible matching concepts, regarding that if two concepts are semantically equivalent, then they must be classified within the same concept type. Concept Importance is a measure of the importance of a concept that shows how centrally located and richly described a concept is in the taxonomy of an ontology. The richly means that the measure must take into account contributions from the concept’s attributes and from all other concepts in the ontology through their relations. Concept Importance is to supply an identifiable link between two heterogeneous descriptions of a concept, regarding that if two ontologies need to be integrated, two concepts in different ontologies should be priorly computed for similarity if their importance-score are most similar than other pairs.

A novel idea of Priorly Matchable Concepts (PMC) propagation method is that direct concept matching is driven between the same Concept Types and Concept Importance score, instead of exhaustive search among all concepts. This means that semantic commonalities between highly heterogeneous concepts can be achieved without taking an exhaustive search in taxonomies. Consequently, PMC propagation method supports content-
based matching in a less complexity and name-base matching avoiding mismatching concepts.

An application of PMC propagation method for developing an effective matching algorithm (called Anchor-Prior) are proposed. The key idea of the called Anchor-Prior algorithm is to start from an Anchor (two matched concepts) to work towards a collection of matched pairs among its neighboring concepts by computing similarities between the priorly collected concepts across the ontologies starting from the anchor. It means that the PMC, which provides additional suggestions for possible matching concepts, is used to determine for which concepts the similarity should be priorly computed.

The method is implemented in Java for matching between OWL ontologies by utilizing Jena OWL API. Experimental results shows that the proposed algorithm presented here performs significantly in terms of accuracy and running compare with some best know methods from OAEI evaluations. Linked Open Data set is also used for schema matching evaluation and acquired better results compare with the existing methods.

1.5 Related Work

In this section some of existing methods and techniques for ontology mapping are summarized. Major mapping methods include large number of area ranging from probabilistic approach, graph based extraction and heuristic to rule based methods. Ontology mapping can be centralized where an upper ontology is shared for different application or decentralized where different kinds of information are mapped.
1.5.1 Lily

Lily is a graph base ontology mapping system [103] (Wang, 2007) extracts semantic subgraph for each entity. It utilizes both structural and linguistic information for initial alignment and then applied subsequent similarity propagation strategy to produce more alignments if necessary. The main function of LILY is to match the heterogeneous ontologies. The matching process in LILY consists of three major modules. The system architecture of LILY is shown in Figure 1.4.

![FIGURE 1.4: System architecture of LILY](image)

1. Extracting semantic subgraph part is responsible for extracting semantic subgraph to represent the real meaning for a given entity in ontology by applying connection subgraphs discovery algorithm.

2. The similarity confidences between entities from different ontologies is computed in Computing alignment module by analyzing
the literal and structural information in semantic subgraph extract in previous part.

3. In Similarity propagation, LILY utilize the satisfactory alignment results decide whether to repeat the process for more alignment.

LILY aims at providing high quality alignments between concept pairs. However, it has number of limitations while aligning the heterogeneous ontologies. The main drawback of LILY is to extract semantic subgraphs from large-scale ontologies. Additionally, complexity is very high in terms of matching time and memory space which degrade the efficiency of system. Time complexity is measured as \( O(k \times N^2) \), where \( N \) is the number of entities and \( k \) is the average time for calculating an alignment. In practical observation, LILY spends four days to get final results of mapping for small size of ontologies.

1.5.2 RiMOM

RiMOM [61] is a multi-strategy ontology alignment framework based on machine learning approach. It uses risk minimization of Bayesian decision to search for optimal mappings from the results of multiple strategies. RiMOM includes the both linguistic and structural strategies for mapping multiple ontology automatically. Overview of RiMOM framework is illustrated in Figure 1.5.

Ontology alignment processes of RiMOM includes five main steps described as follows:

1. Preprocessing: Given two ontologies, RiMOM generates the description for each entity. Then, it calculates the two similarity
2. Linguistic-based ontology alignment: In this step, multiple linguistic-based strategies are executed. Each strategy uses different ontological information and obtains a similarity result for each entity pair. These strategies will be dynamically selected to be included in different alignment tasks.

3. Similarity combination: This step combines the similarity results obtained by the selected strategies. The weights in the combination are determined by the two similarity factors.

4. Similarity propagation: This step considers structural similarity. Three similarity propagation strategies, namely, concept to concept, property to property, and concept to property are used.
5. Alignment generation and refinement: This step fine tunes and outputs the alignment result.

RiMOM comprised different strategies to alignment of multiple ontologies. It achieved good results in both the schema matching and instance matching of ontologies. However, RiMOM still have several limitations in alignment process. Similarity estimations are appropriate for some special situation only, like linguistic similarities are limited to consider the elements that have the same label. Scalability is still a critical problem in instance matching alignment. Execution times and memory requirements increased rapidly in input size grows. For example, RiMOM consumed 36 hours to generate the alignment in some datasets.

1.5.3 ASMOV

ASMOV, which is the automated semantic papping of ontologies with validation [53, 54], is a tool that automates the ontology alignment process by exploiting different features of ontology. ASMOV supports integration of heterogeneous systems using their data source ontologies. It iteratively calculates the similarity between entities for a pair of ontologies by analyzing four features: lexical description (id, label, and comment), external structure (parents and children), internal structure (property restrictions for concepts; types, domains, and ranges for properties; data values for individuals), and individual similarity. The system architecture of ASMOV is depicted in Figure 1.6.

Similarities between each pairs are calculate first and then an overall similarity measure is accomplished and stored in two-dimensional
matrices for each concepts, properties and individuals. A pre-alignment is obtained from these similarity matrices with the highest confidence score. A semantic validation process is run to the pre-alignment to detect the semantic mismatches. After several passes in validation process final alignment is obtained. Though ASMOV shows optimal results in mapping scalability are still needed. It takes 3 hours and 53 minutes in order to generate an alignment.

The main differences between aforementioned algorithms and the Anchor-Prior algorithm are as follows: The PMC propagation method is to directly match between concepts. The possibly matching concepts are driven from the same Concept Types and their Concept Importance-score similarity, instead of exhaustive search among all concepts as above methods mentioned. This means that semantic commonalities between highly heterogeneous concepts can be achieved without taking an exhaustive search in taxonomies. Consequently, PMC propagation method supports content-based matching in a less complexity and name-base matching avoiding mismatching concepts.
1.5.4 Anchor-PROMPT

Anchor-PROMPT (Noy, 2001) [82] is a tool for ontology mapping and merging based on term matching automatically. It takes a set of pairs of related terms (anchors) as an input and produces a set of new pairs of terms which is semantically close.

![System architecture of PROMPT](image)

**Figure 1.7: System architecture of PROMPT**

Anchor-PROMPT traverses the paths between the anchors in the corresponding ontologies to find similar terms. Figure shows the traversing paths between Anchors. For example, in *Figure 1.7*, there are two pairs of anchors in ontology 1 and ontology 2 as classes $A$ and $B$, and classes $H$ and $G$. There exist parallel paths to $A$ to $H$ and $B$ to $G$. It traverse the two paths in parallel, incrementing the similarity score between each
two classes (classes C and D and E and F) that reached in the same step and repeat the process for all the existing paths that originate and terminate in the anchor points for cumulatively aggregating the similarity score.

Evaluations show that Anchor-PROMPT achieved promising precision depending on the size of the initial anchor set and the maximum length of the path that traversed. But it does not work well for all ontologies. For example, if one ontology has a deep concept hierarchy and another has shallow hierarchy with few levels Anchor-PROMPT does not work properly. Additionally, Anchor-PROMPT is time consuming. Worse case running time of Anchor-PROMPT is $O(n^2 \log^2(n))$, which is very high compared to other methods.

1.5.5 Anchor-Flood

Anchor-Flood algorithm [93, 68] aligns two large scale ontologies or one large scale and small scale ontologies effectively. It starts with anchors and gradually explores concepts by collection neighboring concepts which include super concepts, siblings and sub-concepts of certain depth to form a pair of blocks across ontologies. Lexical and structural relations are used to find the similarities among concepts. Figure 1.8 shows the building block of Anchor-Flood algorithm.

The main focus of Anchor-Flood is to solve the scalability problem in aligning the large ontologies. It capable to minimize the comparisons between entities, which reduce the time complexity to performed the alignment. However, Anchor-Flood works well with in upper concept level only, it still takes longer time subsumptions level.
Figure 1.8: System architecture of Anchor-Flood

The main differences between anchor approaches such as Anchor-PROMPT and Anchor-Flood, and the Anchor-Prior are:

- Both previous anchor approach [82, 93, 68] and the Anchor-Prior propagate new matching concepts from an anchor. However, the Anchor-Prior instead of matching to all concepts by traversing taxonomies completely to find initial anchors between ontologies or manually providing initial anchors, considers on finding initial anchors from first three top levels of concept types DC, PC, and RC. The anchor are always high level in the conceptual taxonomy of an ontology. It is very important to explore new matching concepts fast and accurately.

- Anchor-Prior differs from the previous approaches [82, 68] by using PMC for arranging possible concepts in prior sets based on Con-
cept Importance. The prior sets are generated from neighbors of an anchor (see Definition 12). Each prior set is a collection of combinations of pairs of concepts across the corresponding sub-segments of neighbors, such as parents, children, grandchildren, siblings and nephews. The pairs belonging to a prior set are sorted in descending order of the importance-score similarity of each pair based on PMC. For this, the higher possible matching concepts are always checked for similarity before lower ones.

1.6 Thesis Outline

In this chapter, the motivation of the work, the main problem tackled, the objectives, and the overview of approach for solving the problem, are described. The remainder of the thesis is organized as follows:

- Chapter 2 presents the fundamental concepts of ontological engineering in order to introduce the current technical foundation of ontologies.

- Chapter 3 presents the similarity techniques for ontology integration problem.

- Chapter 4 presents Priorly Matchable Concepts propagation Approach. This is one of the main chapters of the thesis. presents an application of PMC for ontology integration. This is to particularise Priorly Matchable Concepts propagation for a specific ontology integration algorithm called Anchor-Prior.

- Chapter 5 consists of data set description, methods, algorithm,
implementation, and its evaluation in terms of complexity for efficiency, as well as in terms of precision and recall for effectiveness.

- Chapter 6 concludes the thesis with a summary of my contributions, advantages and limitations, a brief history of my research progress, and a list of issues and perspectives sketched for the future research.
Ontology Engineering

Ontology engineering has been recently attracted considerable interest. It is a new branch of knowledge engineering which exploits the principles of (formal) ontology to build ontologies.

Ontological Engineering refers to the set of activities that concern the ontology development process (ontology life cycle), and the methodologies, tools, and languages for building ontologies [41].

This chapter describes the current state of the art of ontological engineering. It covers ontology notions, methodologies for ontology building, ontology representation and development tools.

2.1 Basic Ontology Concepts

2.1.1 What is an Ontology?

There are many interpretations about what an ontology is. However, the ontology community has come to a consensus on giving up its definition. The most used ontology definitions are presented as follows:

1. Ontology is a term in philosophy and it is the science of what is, of the kinds and structures of objects, properties, events, processes and relations in every area of reality [94].

2. An ontology is, a formal, explicit specification of a shared conceptualization [42].

3. An ontology is hierarchically structured set of terms for describing
a domain that can be used as a skeletal foundation for a knowledge base [97].

Ontology in the first definition is related to philosophy, it means theory of existence. It tries to explain what is being and how the world is configured by introducing a system of critical categories to account things and their intrinsic relations. The second definition is generally accepted as a definition of what an ontology is for the AI community. A conceptualization refers to an abstract model of some phenomenon in the world which identifies the relevant concepts of that phenomenon. Explicit means that the type of concepts used and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine readable. The last third, ontology is defined as a body of knowledge describing some domain, typically a common sense knowledge domain, using a representation vocabulary as described above. In this case, an ontology is not only the vocabulary, but the whole upper knowledge base. From knowledge-based systems point of view, it is defined as a theory/system of concepts/vocabulary used as building blocks of an information processing system by R.Mizoguchi [71]. For an information system, an ontology is a representation of some preexisting domain of reality which: (1) reflects the properties of the objects within its domain such that there is a systematic correlation between reality and the representation itself, (2) is intelligible to a domain expert, and (3) is formalized in a manner supporting automatic information processing.
2.1.2 The Main Components of an Ontology

An ontology consists of a number of different components. The names of these components differ between ontologies depending on the ontology language used, philosophical persuasion or background of the authors. Despite this, their core components are largely shared between different ontologies.

![Ontology Diagram]

**Figure 2.1: Fragments of ontologies**

Classes or concepts are the main entities of an ontology. These are interpreted as a set of individuals in the domain. They are introduced in OWL by the owl:Class construct. For example, in *Figure 2.1*, *Book* and
Person are classes.

Individuals or objects or Instances are interpreted as particular individual of a domain. These are introduced in OWL by the owl:Thing construct. For example, in Figure 2.1, Artificial Intelligence is an individual.

Relations or object properties are the ideal notion of a relation independently to what it applies. Relations are interpreted as a subset of the product of the domain. These are introduced in OWL by the owl:ObjectProperty or owl:DatatypeProperty construct. For example, in Figure 2.1, creator and topic are relations.

Notice that, properties are roughly equivalent to slots in Protege. They are also known as roles in description logics and relations in UML and other object oriented notions. In GRAIL and some other formalisms they are called attributes.

Datatype properties link an individual to an XML Schema Datatype value or an rdf literal. In other words, they describe relationships between an individual and data values.

Datatypes are particular parts of the domain which specify values as opposed to individuals, however, values do not have identities. For example, in Figure 2.1, String and Integer are datatypes.

2.1.3 Types of Ontologies and their Uses

Ontologies may exist at many levels of abstraction. The ontologies are classified into different broad categories as follows:

- Upper ontology is defined by [88] as a high-level, domain independent ontology, providing a framework by which disparate systems
may utilize a common knowledge base from which more domain-specific ontologies may be derived.

- Core ontology is used by ontologies to describe the most important concepts in a specific domain. A Domain ontology specifies concepts particular to a domain of interest and represents those concepts and their relationships from a domain specific perspective.

- Task ontology [71] is a system of vocabulary for describing the problem solving structure of all the existing tasks domain-independently.

- Application ontology is an application service-specific ontology related to specific pieces of corporate business and human resource information.

- Domain, or subject area, ontology defines the terminology and concepts relevant to a particular topic or area of interest.

- Process ontology defines the inputs, outputs, constraints, relations, terms, and sequencing information relevant to business processes (e.g., ISO PSL - Process Specification Language).

- Interface ontology defines the structure, content, messaging, and other restrictions relevant for a particular interface (e.g., application programming interface (API), database access, CORBA IDL, scripting language).

- Service ontology defines a core set of constructs for describing the vocabularies and capabilities of services.
2.1.4 Ontology Model

We assume a real world \((A, V)\), where \(A\) is a finite set of attributes and \(V\) is the domain of \(A\). Also, \(V\) can be expressed as a set of attribute values, and \(V = \bigcup_{a \in A} V_a\) where \(V_a\) is the domain of attribute \(a\). In this chapter, the following assumptions are made:

Definition 1 (Ontology) An ontology is a quintuplet:

\[ O = (C, \sum, R, Z, I) \]  

(2.1)

where,

- \(C\): is a set of concepts (the classes);
- \(R\): is a set of binary relations between the concepts from \(C\), or between the concepts from \(C\) and the values defined in a standard or user-defined data type;
- \(Z\): is a set of axioms, which can be interpreted as integrity constraints or relationships between instances and concepts. This means that \(Z\) is a set of restrictions or conditions (necessary & sufficient) to define the concepts in \(C\);
- \(I\): is a set of instances of concepts belonging to \(C\).
- \(<C, \sum>\): is the taxonomic structure of the concepts from \(C\) where \(\sum\) is the collection of subsumption relationship (\(\sqsubseteq\)) between any two concepts from \(C\). For two concepts \(c_1, c_2 \in C\), \(c_2 \sqsubseteq c_1\) if and only if any instances that are members of concept \(c_2\) are also members of concept \(c_1\), and the converse is not true.

\(R\) is known as a set of properties. For every \(p \in R\), there is a specific domain \(D\) and range \(R\) such that \(p : D \rightarrow R\), where \(D \subset C\), and if \(R \subset C\)
then $p$ is called an object property, otherwise, if $R$ is a set of standard or user-defined data types, then $p$ is called a data type property. Assume that concepts $c$ and $c'$ correspond to the domain and range of property $p$, respectively, where $p$ is also known as an attribute of concept $c$. There are two given instances $v$ and $v'$ that belong to the corresponding concepts $c$ and $c'$ respectively. Denote $vR^pv'$ as the relation from instance $v$ to $v'$ based on the property $p$, and the relation from instance $v'$ to $v$ based on the property $p$ is denoted as $vR^-p v'$.

**Definition 2 (Concept)** A concept $c$ of an $(A, V)$-based ontology is defined as a triplet:

$$c = (z_c, A^c, V^c)$$

where $c$ is the unique identifier for instances of the concept. $A^c \subseteq A$ is a set of attributes describing the concept and $V^c \subseteq V$ is the attributes' domain: $V^c = \bigcup_{a \in A^c} V_a$. $z_c \subseteq Z$ is the set of restrictions or conditions (necessary & sufficient) to define the concept $c$. The $z_c$ can be represented as a constraint function $z_c : A^c \rightarrow Z$ such that $z_c(a) \in Z$ for all $a \in A^c$.

Pair $(A^c, V^c)$ is called the possible world or the structure of the concept $c$. Notice that within an ontology there may be two or more concepts with the same structure. If this is the case, then the constraint function $z_c$ is useful for expressing the associated relationships. For example, two concepts RedWine and WhiteWine have the same structure $\{\text{hasMaker}, \text{hasColor}\}$. But $z_{\text{RedWine}}(\text{hasColor}) = \{\exists \text{hasColor} = \text{red}\}$ and $z_{\text{WhiteWine}}(\text{hasColor}) = \{\exists \text{hasColor} = \text{white}\}$.

**Definition 3 (Attribute)** Attribute $a$ is defined as a triplet:

$$a = (L_a, S_a, f_a)$$

where
• *a* is the unique identifier for instances of the attribute.

• *La* is a set of atomic elements

• *Sa* is semantic of the attribute *a*

• Function *fa*: *I* \(\rightarrow\) *Va*

**Example 1** The attribute of *Age* defined as follows: 
\[ L_a = \{t_1 \leq 200, t_2 \geq 18, t_3 < 18, \ldots\}, S_a = t_1 \land t_2 \].
The definition says that \(18 \leq Age \leq 200\).

### 2.2 Methodologies for Ontology Building

#### 2.2.1 Criteria for Designing Ontology

According to [57, 48], the criteria clarity, coherence consistency, extensibility and minimal ontological commitment [102] are applied to guide and evaluate the whole process of ontology design.

• **Term Clarity**: It is important to define terms clearly without ambiguity for every class and property in system design. Every part should have the same semantic meaning, which also means sharing the same vocabulary, because all participants need to clearly understand any communication that occurs between them. For example, the term *instructor* has the same meaning as *teacher* or *professor* and these cases need to be clarified specifically and clearly whether in class or property.

• **Coherence consistency**: In this case, ontological statements must have consistency even when natural language is used. For example, logical consistency must be applied in formal axioms.
• Extensibility: This means that if any parts of the ontology structure need to be extended or changed, these changes should be customized without any need to revise existing definitions.

• Minimal ontological commitment required: The designer responsible for a vocabulary should consider problems such as the fact that the vocabulary might contain needless and redundant classes and properties in both general and very specific models. The structure of the ontology must reflect only the intended domain and few claims should be made to avoid restricting future extensions.

2.2.2 Engineering-oriented Method

Most ontologies are developed via an engineering-oriented method as a small group of engineers carefully builds and maintains a representation of their view of the world. Maintaining such large ontologies in an engineering-oriented manner is a highly complex process where developers need to regularly merge and reconcile their modifications to ensure that the ontology captures a consistent and unified view of the domain. One example of these tools is Protege, which is used by Stanford University for knowledge acquisition. It provides a graphical and interactive ontology-design and knowledge-base development environment. Ontology developers can access relevant information quickly, and navigate and manipulate the ontology.
2.2.3 Collaborative-oriented Method

Overview

Currently, there are several tools oriented towards collaborative work [100, 92, 57, 1, 96, 35] as a consensus-building mechanism that allows a large group of people to contribute or annotate a common ontology in a collaborative manner. The authors of [100] have developed Collaborative Protege as an extension to the client-server Protege. Collaborative Protege allows entire groups of developers who are collaboratively building an ontology to hold discussions, chat, and make annotations and changes as a part of the ontology-development process. OntoWiki [1] is a web-based ontology which focuses on an instance editor that provides only rudimentary capabilities such as the history of changes and ratings of ontology components. OntoEdit [96] is a CoO editing environment that integrates numerous aspects of ontology engineering and allows multiple users to develop ontologies. KAON [35] focuses on changes of ontology that can cause inconsistencies, and proposes deriving evolution strategies in order to maintain consistencies.

According to Holsapple [48], the collaborative approach phases to ontology design are:

- The preparatory phase that defines the criteria for ontology design, specifies boundary conditions for the ontology, and determines standards for assessing its success.

- The anchoring phase that includes development of the initial version of the ontology that will feed the next phase (evaluation phase) based on compliance with the design criteria.
• The iterative improvement phase that enhances the ontology until all participants’ viewpoints reach a consensus through a collaborative building technique. In this phase, the ontology will be revised and evolve its structure by collaboration of the participants. At each iterative improvement, the ontology is evaluated by the aforementioned standards and conditions.

• The application phase that demonstrates the use of CoO by applying it in various ways.

Agreeing with Holsapple, the authors of [57] follow the above phases in which they start by defining the criteria for ontology design, by applying the ontology-building steps described in [82] to design the initial ontologies. Their collaborative methodology for ontology-building supports a team effort to iteratively revise and evolve the initial ontology until all participants’ understanding of the ontology reaches consensus. The consensus is achieved through voting in a Nominal Group Technique manner [36].

The Collaborative ONTology ENgineering Tool (ContentCVS) is a system that is available for download as a Protege plugin [92]. The tool supporting collaboration provides the means for: (1) keeping track of changes of ontology versions, (2) identifying conflicts between the versions of the ontology, (3) constructing a reconciled ontology from conflicting versions and identifying errors, and (4) suggesting possible ways to repair the identified errors with minimal impact on the ontology. The aforementioned methods agree that CoO involves a group of people contributing to an ontology in a specific domain of interest. CoO allows an entire group to participate in the process of ontology-building
by reaching a consensus, and usually aims at completeness. CoO-building involves individuals contributing to understanding of their ontological perspective, but everybody works together to build the ontology. The purpose of CoO is to generate the best representative ontology from various versions of an ontology. The final ontology must be best reflect the conflicting versions of the ontology in a compromise.

However, the above approaches just focus on human collaboration to build a common ontology. Differing from the previous approaches, the main goal of this research is to investigate the techniques that support a solution to conflicts among different participants’ viewpoints in the collaborative ontology (CoO) process. A machine is considered as a leader of the collaborative group, via which, conflicts among the versions of the ontology are identified, and a reconciled version that best reflects the conflicting versions in a compromise is generated automatically. The main contributions of this research are as follows:

Criteria for Collaborative Ontology

In this section, the criteria guiding throughout the whole process and CoO evaluation are presented. These criteria are inclusive, egalitarian, interactive, representative, reconcilable, trust and proof.

- Inclusive: The problem is how to reach consensus if a few people take part in a collaborative process in which their opinions are different or even contradictory. To overcome the problem, as many contributors/participants as possible should be involved in the CoO process. The number of participants should be large enough to reach consensus. Usually, the collaborative process needs to last
long enough to welcome many participants and allow them to learn to work in a collaborative manner.

- **Egalitarian:** Participants should be afforded as much opportunities for collaboration as possible, e.g. the more competent an expert is in their own field the more opportunities for collaboration they should receive. CoO must consider this criterion to assign participants permissions suited to their majors. Today, CoO systems such as Wikipedia, ContentCVS, and OntoWiki commonly allow anyone to change an ontology freely, even though participants might lack knowledge of the relevant field.

- **Interactive:** Any interaction between the group members can involve more or less collaboration (e.g. negotiation has a stronger collaborative flavor than giving instructions). In fact, many ontology development projects today set boundaries on what different users can do. For instance, some users can take proposals for changes but not make the changes themselves. Others can comment on these proposals, but not create new ones. Another group of users can affect the changes in the ontology based on the discussion. Still others can perform quality control by reviewing and approving the changes.

- **Representative:** Participants in an effective consensus process should strive to reach the best possible decision for the group and all of its members in order to generate the best representative ontology from the given versions, rather than opt to pursue a majority opinion, potentially to the detriment of a minority.
• Reconcilable: CoO should be a good compromise and be acceptable to the conflict participants. An effective CoO strives to capture consensus over differences using compromise and other techniques to avoid or resolve mutually exclusive positions within the group.

• Trust: Notice that one of the features that all CoO systems need today is trust. Any user with writing privileges can edit anything in an ontology. However, users need to have more fine-grained control, particularly in the development of large ontologies. For example, users with expertise in an area represented by some part of an ontology should be able to edit that part, but may only be able to browse other parts or link to them.

• Proof: Determining whether a change made to an ontology by some participant is correct or not. Even when the change is inconsistent with other versions, the change must be tracked and the conflict must be identified. Until all participants’ viewpoints reach consensus, a reconciled ontology is constructed from all participants’ shared understanding of the ontological perspective.

Consensus for Collaborative Ontology Approach

In the following, the features of the proposed consensus methodology for CoO are briefly presented.

**Phase 1 Preparatory.** The criteria for ontology building are provided. The criteria are aimed at guiding and evaluating the CoO process. In this phase, core ontologies are also introduced to participants.
• Criteria Design: The criteria are designed including clarity, coherence, consistency, extensibility, minimal ontological commitment, identity and type of concept for ontology design. These criteria are important in both guiding development of the ontology and evaluating the degree of its success. It also provides seven criteria as composites of inclusive, egalitarian, interactive, representative, reconcilable, trust and proof for CoO. These criteria are important in guiding the development of CoO systems and assessing the quality of the CoO process.

• Core Ontology Design: Core ontologies are ontologies describing the most important concepts of a specific domain of interest. Participants contribute to and annotate a core ontology iteratively to generate new versions of the ontology. Concepts in core ontologies should permit any changes as extensions for customization without the need of revising existing definitions. To develop the core ontology, the following ontology-building steps described in [82] are applied:

  – Step 1. Determine the domain and scope of the ontology.
  – Step 2. Consider reusing existing ontologies.
  – Step 3. Enumerate important terms in the ontology.
  – Step 4. Define the classes and the class hierarchy.
  – Step 5. Define the object properties and date type properties of classes
  – Step 6. Define the restrictions of the data type and the object properties
Phase 2 Contribution. Processes occur in rounds, allowing individuals to contribute or change their opinions of the current version of the ontology. First, a core ontology is a changeable ontology in that the server only accepts changes to it. Participants can access the server to check out the changeable version and make a copy of the version as their own target version, which they can modify at will.

Phase 3 Keeping track of changes and conflicts. The computer receives an updated version of the ontology from a participant. It analyzes the updated version by keeping track of changes of the ontology, and identifies conflicts between the updated version and the other ones.

- Keeping track of different versions and changes to an ontology: Participants can commit their changes to the server at any time. This allows several participants to make concurrent changes to a changeable ontology via their own target ontology. Participants can share their understanding of the perspective of the target ontology, which represents the change history and the target ontology. The target versions are kept in the server’s shared repository as new generated versions.

- Identifying conflicts: The conflicts on the body of a target ontology, and conflicts between the target ontology and generated versions are addressed. Compare the target version and generated versions, then identify conflicts among them.

Phase 4 Controlled feedback. If no consensus is reached, the integration of previous versions will be used as a new version of the ontology,
and the computer will show this new version of the group’s contribution (indicating to each individual their own previous response in Delphi) to each participant in the group. Phase 2 is repeated until consensus is reached.

2.3 Ontology Representation Languages and Development Tools

2.3.1 Ontology Representation Languages

Ontology Evolution

Mid-90s, Extensible Markup Language (XML) was developed from GSML (Standard generalized Markup Language (SGML ISO 8879: 1986)) to provide a set of metadata tags expecting representing Semantic Web data, but XML does not define the meaning of the tags. Thus, available information on the Web can only be accessed with syntactic interoperability. Software agents can not understand and process information effectively. In the late 1990s, the World Wide Web Consortium (W3C) Metadata Activity started work on RDF Schema (RDFS), a language for RDF vocabulary sharing. The RDF became a W3C Recommendation in February 1999, and RDFS a Candidate Recommendation in March 2000. In 2001, Tim Berners-Lee and colleagues have established a vision of the Semantic Web [4] that is an extended Web of machine-readable information and automated services that extend far beyond current capabilities. The explicit representation of the semantic underlying data, programs, papers, and other Web resources will become a knowledge-based Web that improves in their capacity to assist human and machine to be more well-collaborative works. In 2003, Description Logic was proposed by
Baader [2], which was considered as knowledge representation in Semantic Web. From which, Description logic are interesting as backbone for knowledge representation and reasoning on Semantic Web. In 2004 (as part of a wider revision of RDF) RDFS became a W3C Recommendation. Though RDFS provides some support for ontology specification, the need for a more expressive ontology language had become clear.

In 2005, at the OWL Experiences And Directions Workshop a consensus formed that recent advances in description logic would allow a more expressive revision to satisfy user requirements more comprehensively whilst retaining good computational properties. In December 2006, the OWL1.1 Member Submission was made to the W3C. The W3C chartered the OWL Working Group as part of the Semantic Web Activity in September 2007. In April 2008, this group decided to call this new language OWL2, indicating a substantial revision. OWL 2 became a W3C recommendation in October 2009. OWL2 introduces profiles to improve scalability in typical applications.

**RDF-Resource Description Framework**

The Resource Description Framework (RDF), which is an a standard model for data exchange on the Web, enables the encoding, interchange and reuse of structured metadata. RDF is an application of XML that extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a triple). This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes. This graph view is the easiest
possible mental model for RDF and is often used in easy-to-understand
visual explanations. Using this simple model, it provides a means for
publishing both human-readable and machine-processable vocabularies
designed to encourage the reuse and extension of metadata semantics
among disparate information communities.

**RDF Schema-RDFS**

RDF-Schema or RDFS The Resource Description Framework Schema
(RDFS or RDF Schema) is an extension of RDF vocabulary to describe
taxonomies of classes and properties. Classes (concepts) are arranged
hierarchically, and the use of properties can be constrained to members
of certain classes. The root of the class hierarchy is rdfs:Resource, rdfs:Class
is subclass of rdfs:Resource. It offers a set of language features with well-defined semantics. Meaning of features is given in mathematical terms.
Additionally, provides mechanisms for describing groups of related re-
sources and the relationships between these resources. RDF Schema
vocabulary descriptions are written in RDF. These resources are used to
determine characteristics of other resources. RDF Schema defines con-
structs for specifying which classes, predicates and individuals exist in
the vocabulary. It specifies how concepts may relate to one another.

**OWL (DAML+OIL)**

Web Ontology Language (OWL) is a language to describe the classes
and relations between them in the web ontologies. OWL extends RDFS
syntax and formalizes its semantics in terms of Description Logics. It
provides much more expressive set of features for enhancing RDF data
with formal semantics.

OWL is an ontology language which is logically sound and complete and at the same time implementable. It allows for more advanced inferences than over RDF with RDFS. OWL is a logical knowledge model which uses RDF/XML syntax for representation.

There are three species of OWL such as OWL-Full, OWL-DL and OWL-Lite. OWL-Full is very expressive, no restrictive on how/where language constructs can be used. It partially support the entailment rules. OWL-DL restricted version of OWL-Full. It corresponds to description logic. Certain restrictions on how/where language constructs can be used in order of guarantee decidability. OWL-Lite is a subset of OWL-DL. Supports primary classification and simple constraint features.

2.3.2 Ontology Development Tools

Incorporating the methodologies and languages, there have been developed many environments for ontology development. Among them, this section takes up OntoEdit [96], WebODE [8], and Protege\(^1\) which cover a wide range of ontology development process rather than being a single-purpose tool which should be covered elsewhere.

**OntoEdit**

OntoEdit [96], professional version, is an ontology editor that integrates numerous aspects of ontology engineering. Two tools, OntoKick and Mind2Onto, are prepared for supporting the phase of ontology capture. OntoKick supports for the collaborative generation of requirements

\(^{1}\)http://protege.stanford.edu/
specifications for ontologies. Collaborative process divided into two processes. Firstly, OntoKick allows for describing important meta-aspects of the ontology. It guides the engineering team stepwise through all relevant aspects and stores these descriptions along with the ontology definitions. Mind2Onto is a graphical tool for capturing informal relations between concepts. Mind2Onto targets at integration of brainstorming processes to build relevant structures of the semi-formal ontology description. It is easy to use because it has a good visual interface and allows loose identification of relations between concepts. However, it is necessary to convert the map into a more formal organization to generate ontology.

**WebODE**

WebODE [8] is a scalable and integrated workbench for ontology engineering that covers and gave support to most of the activities involved in the ontology development process (conceptualization, reasoning, exchange, etc.) and supplied a comprehensive set of ontology related services that permit inter-operation with other information systems. WebODE provides high extensibility in an application server basis, allowing the creation of Middleware services that will allow the use of ontologies from applications. Ontologies built with WebODE can be easily integrated with other systems by using its automatic exportation and importation services from and into XML, and its translation services into and from varied ontology specification languages such as RDF(S), OIL, DAML+OIL X-CARIN and F-Logic. Like OntoEdit, WebODEs ontology editor allows the collaborative edition of ontologies. The collaborative
edition of ontologies is ensured by a mechanism that allows users to establish the type of access of the ontologies developed, through the notion of groups of users. Synchronization mechanisms also exist that allow several users to edit the same ontology without errors. Similar to OntoEdit, WebODE has Ontoclean methodology [44] to build a theoretically correct ISA hierarchy.

**Protege**

Protege provides a suite of tools to construct domain models and knowledge base. It supports the creation, visualization, and manipulation of ontologies in various representation formats. Protege is powerful platform in the use phase of ontology: Use for knowledge acquisition, merging and alignment of existing ontologies, and plug-in new functional modules to augment its usability. It has been used for many years for knowledge acquisition of domain knowledge and for domain ontology building in recent years. The Protege platform provides two categories of modeling ontologies. One is Protege-Frames editor where user can construct ontologies as a set of classes organized in a subsumption hierarchy and a set of slots describe the properties and relationship about classes and set of instances of those classes. Another category is Protege-OWL editor; enable to construct ontologies based on OWL (Web Ontology Language) including descriptions of classes, properties, instances and reasoning.
Chapter 3

Similarity Techniques for Ontology Integration

3.1 Basic Techniques

The importance matter of ontology integration is finding the relations between entities expressed in different ontologies. Very often, these relations are equivalence relations that are discovered via the similarity measure between the ontological entities. According to the aforementioned studies, the techniques of similarity analysis that have been explored for ontology integration can be classified into following four groups:

- **Instance-based similarity**: The similarity between two concepts is based on common instances.

- **Lexical-based similarity**: The similarity between two concepts is based on analysis of the linguistic meanings of associated names.

- **Schema-based similarity**: The similarity between two concepts is based on analysis of the similarity between associated properties.

- **Taxonomy-based similarity**: The similarity between two concepts is based on analysis of their structural relationships, such as subsumption.

However, the above techniques are generated from the word similarity measure. A formal definition of the similarity method is as follows: let
$x$, $y$, $z$ denote words, and $\text{sim}(x, y)$ represent the semantic similarity between $x$ and $y$.

- $\text{sim}(x, y) \in [0, 1]$

- if $\text{sim}(x, y) = 1$ then $y = x$ or $x$ is semantically equivalent to $y$.

- $\text{sim}(x, y) = 0$: two words are disjoint, i.e., no common characteristics.

- $\text{sim}(x, y) = \text{sim}(y, x)$: similarity is symmetric

- $\text{sim}(x, x) \geq \text{sim}(y, x)$: similarity is maximality

In this section presents some methods to measure the similar degree between two words and two strings (names/labels/comments) which are often used to name or describe the entities of an ontology.

### 3.1.1 String-based Similarities

String-based similarity techniques are often used in order to match names and name descriptions of ontological entities. These techniques consider strings as sequences of letters in an alphabet. They are typically based on the following intuition: the more similar the strings, the more likely that they denote the same concepts.

**Name Co-reference**

Entity’s name co-reference is determined when identified names in the text refer to the same entity. Most entities names exist in documents that have different forms. An entity’s name co-reference is based on the following criteria:
• Suffix takes as input two strings and checks whether the first string ends with the second one (e.g., Compact Disc = CD).

• Prefix takes as input two strings and checks whether the first string starts with the second one (e.g., Net = Network).

• Aliases within the document that are indicated by parenthetical (e.g., KFOR = Kosovo Force)

• Variants of a name (e.g., W. Bush = George W. Bush)

A prefix or suffix pre-similarity can be defined from the prefix and suffix tests, which test whether one string is the prefix or suffix of another. Prefixes, suffixes or aliases can be recognized from a text corpus via lexico-syntax patterns such as parenthetical. Very often, this model is examined under learning approaches.

**Edit Distances Similarity**

The edit distance, $e(x, y)$, from a string $x$ to a string $y$ is the minimum number of simple edit operations (insert, delete, replace, transpose) required to transform one string into the other. Edit distances were designed to measure the similarity between strings that may contain spelling mistakes. The edit distance similarity can be computed as follows:

$$sim(x, y) = 1 - \frac{e(x, y)}{\max(size(x), size(y))}$$

(3.1)

**Common Substring Similarity**

The $z$ is denoted as the longest common substring between $x$ and $y$ starting from the first character of both and containing at least three
characters. The common substring similarity is defined as follows:

\[ \text{sim}(x, y) = \frac{2 \cdot \text{length}(z)}{\text{length}(x) + \text{length}(y)} \] (3.2)

It is easy to see that this measure is indeed a similarity. One could also consider a subsequence similarity as well. This definition can be used for building functions based on the longest common prefix or longest common suffix.

### 3.1.2 Language-based Similarity

Language-based similarity considers names or express names (label or comment) as words in some natural language, e.g., English. They are based on natural language processing techniques exploiting morphological properties of the input words. They are applied to the names of entities before running string-based or linguistics-based techniques in order to improve their results. This approach is called pre-processing, which is shown in Figure 3.1.

![Figure 3.1: Language-based similarity process](image)

- **Tokenization**: The concept labels are segmented into sequences of tokens by a tokeniser which recognises punctuation, cases, blank
characters, digits, etc. For example, \textit{has Hands Free Kits} becomes tokens\{\textit{has, Hands, Free, Kits}\}.

- Stemming: The strings underlying tokens are morphologically analyzed in order to reduce them to normalised basic forms. For example, tokens \{\textit{has, Hands, Free, Kits}\} becomes root form \{\textit{have, hand, free, kit}\}.

- Stop word: Stop word is a word which is irrelevant to the content of the text, such as a pronoun, conjunction, preposition, article, and auxiliary verb. Removing stop words ensures more efficient processing by downsizing and the remaining words are relevant to the content. For example, after removing the stop words of the set \{\textit{have, hand, free, kit}\}, it becomes \{\textit{hand, free, kit}\}.

3.1.3 Linguistic-based Similarity

Linguistic-based similarity (lexicons or domain specific thesauri) is used in order to match words based on their linguistic relations\(^1\), e.g., synonyms, hyponyms as several works \cite{26, 27}. A lexical resource such as the lexical database WordNet is particularly well suited for similarity measures, since it organizes nouns and verbs into hierarchies of is-a relations, and concepts can be related based on the path lengths between them such as in Lch \cite{59} and Wup \cite{105}. In addition, WordNet has extremely fine-grained notions of word sense, which precisely capture even minor distinctions between different possible word senses, thus the similarity measures can based on the information content of the concepts.

\(^1\)In this case the names of ontology entities are considered as words of a natural language.
Moreover, concepts can be related in many ways beyond mere similarity to each other. For example, a wheel is a part of a car, night is the opposite of day, snow is made up of water, a knife is used to cut bread, etc. As such WordNet provides additional (nonhierarchical) relations such as has-part, is-made-of, is-an-attribute-of, etc.

3.2 WordNet-based Similarity

The lexical database WordNet is particularly well suited to similarity measures, since it organizes nouns and verbs into hierarchies of is-a relations. In version 2.0, there are nine noun hierarchies comprising 80,000 concepts, and 554 verb hierarchies comprising 13,500 concepts.

Many previous works focus on finding the similarity between words based on WordNet. However, they can be distinguished two basic approaches: (1) The similarity measures are based on the path lengths between concepts such as in Lch [59] and Wup [105]. Most of these similarity measures are subject to the is-a hierarchy containing the concepts. But is-a relations in WordNet do not cross part-of-speech boundaries, so these WordNet-based similarity measures are limited to distinguishing between noun pairs (e.g., cat and dog) and verb pairs (e.g., run and walk). While included in WordNet, the adjectives and adverbs are not organized into is-a hierarchies. (2) The similarity measures are based on information content, which is a corpus-based measure of the specificity of a concept. These measures include Res [91], and Jcn [56]. Intrinsic to the calculation of information content is the use of tagged corpora. It is expected that the more often a concept appears in a corpus, the less
specific it is, so the methods depend on tagged corpora. Such a strategy is not without a downside; there are two well known deep problems. Manually tagging corpora is monotonous and very time consuming; It is very difficult to obtain a statistically valid and reliable corpus that accurately reflects the word usage; many relatively common words may not appear, even in a very large corpora. This problem is usually referred to as the sparse data problem.

However, concepts can be related in many ways beyond mere similarity. For example, the words that occur together in a synset have the synonym relation. For example, the words learner occurs in two noun synsets \{learner, scholar, assimilator\} and \{apprentice, learner, prentice, student\} occurs in two noun synsets \{student, pupil, educate\} and \{scholar, scholarly person, bookman, student\}. Thus, scholar is a common word of a student’s and learner’s synset, so student and learner have a relation. If continue finding synonyms of words in student’s and learner’s synsets, the number of similar words that occurs together with student and learner may be much larger. Thus the similarity degree between student and learner is much larger. Moreover, each word can occur in many synsets that cross-part-of-speech. For example, the word base occurs in seven adjective synsets, three verb synsets, and 19 noun synsets, thus the similarity crosses-part-of-speech. Thus, the following formula is proposed for measuring the semantic similarity of words:

\[
\text{sim}(w_1, w_2) = \max_{\text{level} = 1, \ldots, n} \left( \frac{\Delta + \sum_{w_i \in \text{Syn}_1 \cap E} \sum_{w_j \in \text{Syn}_2 \cap E} \text{Inc}(w_i, w_j)}{\min(\text{size(Syn}_1), \text{size(Syn}_2)) + \text{size}(E)} \right)
\]  (3.3)

where

\[
\text{Inc} = \begin{cases} 
0 & \text{if } w_i \neq w_j \\
1 & \text{if } w_i = w_j 
\end{cases}
\]
If \( Inc(w_1, w_2) = 1 \) then \( E = E \cup \{w_1\} \).

\( \triangle \) is the total return value of \( Inc \) at level \( k \), \( k \) is the current level.

When the level is increased from 1 to \( n \), each iteration then,

\[ Syn_1 = \bigcup_{w \in Syn_1 \cap E} Synonym(w) \quad \text{and} \quad Syn_2 = \bigcup_{w \in Syn_2 \cap E} Synonym(w). \]

We experimented with the method to determine the similarities between 100 pairs of words with different similarity degree and crossing-part-of-speech. When choose a level of three, and the limit of the size of array Syn is 1000, most of the similarities between words are determined. The higher the level, the greater the similarities between words. For example, the similarity between learner and student at level 1, 2, 3 is 0.24, 0.65, 0.84, respectively.

### 3.3 Lexico-Syntactic Patterns-based Similarity

#### 3.3.1 Lexico-Syntactic Patterns for Hypernym

Three syntactic phenomena commonly encode the hypernymic proposition: verbs, appositive structures, and nominal modifications. Here, the appositive structures [47, 46, 76] are considered in which two noun phrases must be contiguous. Three types of appositive cues can then mark the second noun phrase: commas, parentheses, or lexical items (including, such as, particularly, and especially). For instance, consider a sentence (*) We identified the activities involving students such as seminars, discussion, conferences., where activity is the hypernym of seminar, discussion, conference. The sentence is then transformed into the following lexico-syntactic expression:

(1a) \( NP_0 \) such as \( NP_1, NP_2, \ldots \), (and \( \parallel \) or) \( NP_i i \geq 1 \), where \( NP_i \) is phrase noun i, are such that they imply
(1b) for all $NP_i, i \geq 1$, hypernym($\text{head}(NP_0), \text{head}(NP_i)$) where $\text{head}(NP_i)$ is head noun of $NP_i$.

In [47, 46, 76] the authors presented most of the lexico-syntactic patterns for hyponymy. However, they have not analyzed the NP to identify its head noun, so the mistakes are common when finding the hypernym relation between concepts. For instance, student is the hypernym of seminar, discussion, conference often acquired from sentences as (*). Therefore, the grammatical noun phrase is analyzed to identify its head noun, which is useful to avoid wrong relations as in the aforementioned instance.

### 3.3.2 Noun Phrase Analysis

Noun phrases normally consist of one or more in functional positions: HEAD (H:), DETERMINER (D:), POST DETERMINER (POD:), MODIFIER (M:), and POST MODIFIER (PM:), where

1. **HEAD**: the noun, the personal pronoun, the demonstrative pronoun, and the possessive pronoun.
2. **DETERMINER**: the definite article, indefinite articles, demonstrative articles, and possessive articles.
3. **POST DETERMINER**: the cardinal numerals, ordinal numerals, and general ordinals.
4. **MODIFIER**: adjective and noun.
5. **POST MODIFIER**: the post modifier is a prepositional phrase which has two functional positions within it, the RELATER (R:) position and the OBJECT OF A PREPOSITION (OP:) position. The RELATER position is occupied by a preposition, for example, *from* in *from the library*, and the OBJECT OF A PREPOSITION position is typically occupied by
a noun phrase, for example *the library in from the library*.

It must be noted that only HEAD (H:) appears in all of the 16 patterns (see Table 3.1) thus a head must appear in a noun phrase. However, all of the other functional labels appear in some patterns but not in others. This means that those functional positions are optional in the noun phrase. This is showed in (**) by placing the abbreviations for those functional positions in parentheses. Any functional position in parentheses may or may not occur in a given noun phrase. If it does occur, it must be in the order indicated by the pattern. For example, DETERMINERS are always first and POST MODIFIERS are always last; POST DETERMINERS follow any co-occurring DETERMINERS and precede any co-occurring MODIFIERS.

(**): noun phrase form: (D:) + (POD:) + (M:) + H: + (PM:)

(the) (three) (large) books (from the library)

Here is a grammatical analysis of the noun phrase *the three large books from the library* based on (**):

- DETERMINER definite article (the)
- POST DETERMINER cardinal numeral (three)
- MODIFIER adjective (large)
- HEAD noun (books)
- POST MODIFIER prepositional phrase (from the library)

### 3.3.3 Lexico-Syntactic Patterns for Similarity Measure

Most instances of a concept are the set of hyponyms of the concept. For example, when the concept *Country* has instances as *Vietnam, Korea,*
Table 3.1: The 16 patterns of noun phrase

<table>
<thead>
<tr>
<th>H: books</th>
<th>D: + POD: + M: + H: + PM: the three large books from the library</th>
</tr>
</thead>
<tbody>
<tr>
<td>D: + H: the books</td>
<td>POD: + M: + H: + PM: three large books from the library</td>
</tr>
<tr>
<td>POD: + H: three books</td>
<td>D: + M: + H: + PM: the large books from the library</td>
</tr>
<tr>
<td>M: + H: large books</td>
<td>D: + POD: + H: + PM: the three books from the library</td>
</tr>
<tr>
<td>D: + M: + H: the large books</td>
<td>M: + H: + PM: large books from the library</td>
</tr>
<tr>
<td>D: + POD: + H: the three books</td>
<td>POD: + H: + PM: three books from the library</td>
</tr>
<tr>
<td>POD: + M: + H: three large books</td>
<td>D: + H: + PM: the books from the library</td>
</tr>
<tr>
<td>D: + POD: + M: + H: the three large books</td>
<td>H: + PM: books from the library</td>
</tr>
</tbody>
</table>
Thus, a method to compute the similarity between two concepts via their instances has been proposed as follows:

\[ L_c = \{l_1, l_2, \ldots, l_n\} \]

is the name/label of the instances of concept \( c \).

\[ A_i = \{a_1, a_2, \ldots, a_k\}, a_j \]

is the set of tokens resulting from two processes: demarcating and possible classification of sections of each string \( l_i \in L_c \) and determining the root form of each token. For example, parsing the name \textit{Hands Free Kits} into tokens \{hand, free, kit\}.

\( G_i = \{g_{1i}, g_{2i}, \ldots, g_{mi}\} \)

is the set of more general words of \( a_j \in A_i, j = 1, 2, \ldots, k \).

Each of the words belonging to \( G_i \) are generated from the \textit{lexico-syntac} patterns.

\[ H = (h_1, h_2, \ldots, h_k) = \bigcup_{i=1}^{n} (G'_i) \]  

(3.4)

where \( G'_i \subseteq G_i \) and if \( h_j \in G'_i, h_j \) exists at least \( \frac{1}{2} n \) sets \( G_i, i = 1, 2, \ldots, n \)

The feature vector of \( H \) is denoted as follows:

\[ \vec{S}_H = (w_1, w_2, \ldots, w_k) \]  

(3.5)

where

\[ w_i = \frac{f_i}{\sum_{j=1}^{k} f_j} \]  

(3.6)

\( f_i \) is number of occurrences of \( h_i \) in the sets \( G_i, i=1..n \).

We define the similarity between two concepts \( c \) and \( c' \) as follows:

\[ \text{sim}(c, c') = \text{sim}(\vec{S}_H, \vec{S}'_H) = \frac{\sum_{(h_i, h'_i) \in K} (w_i \cdot w'_i)}{\sqrt{\sum_{i=1}^{n} (w_i)^2} \cdot \sqrt{\sum_{i=1}^{n} (w'_i)^2}} \]  

(3.7)

where \( K = \{(h_i, h'_i) | \text{sim}(h_i, h'_i) = 1\} \)
3.4 Collaborative Acquisition Algorithm

While comparing a result of a relation between two concepts \( c_1 \) and \( c_2 \) denoting \( R(c_1, c_2) \) for WordNet, there are three possibilities:

1. Both concepts \( c_1 \) and \( c_2 \) are in WordNet, and their relation \( R(c_1, c_2) \) is already in the database of WordNet; updating ontology integration tasks is suggested in [27].

2. Both concepts \( c_1 \) and \( c_2 \) are in WordNet, and the relation \( R(c_1, c_2) \) is not; updating WordNet is suggested.

3. The concepts \( c_1 \) and \( c_2 \) are not present; adding these concepts and the corresponding \( R(c_1, c_2) \) relation to the Knowledge of Assistant WordNet is suggested.

![Figure 3.2: The acquisition process](image)

In collaborative acquisition algorithm, which combines WordNet and text corpus to discover new relations between ontological entities
for ontology integration tasks, is sketched as follows (see Figure 3.2):

- Knowledge of Assistance WordNet is a Concept Net based on the ontology with the following relations: is kind of, is equivalent of. It receives messages from the Feedback component, then updates the relations between the entities of ontologies which are not WordNet-based.

- Mining from Text Corpus is the procedure that is mentioned in the above section. It discovers new relations between ontological entities via the text corpus.

- Ontology Integration Task is presented in the work [27]. It receives the relation \( R(c_1, c_2) \) and updates OnConceptSNet.

- Feedback is a cache of the new relation and mark (mark is used to identify the new relation which should be updated in Knowledge of Assistance WordNet or WordNet-based.

3.5 Ontological Similarities

3.5.1 Name Similarity

Given two concepts \( c \) and \( c' \), the similarity between these concepts \( c \) and \( c' \) can be defined as follows:

\[
sim(c, c') = \sim(\text{Name}_c, \text{Name}_{c'})
\] (3.8)

3.5.2 Annotation Similarity

The annotation similarity considers ontological entities independently of their relations with other entities or their instances. The similarity between entities is determined by the similarity between their names,
labels, comments and annotations. It is computed by means of the feature vectors of the two concepts [69].

The feature vector of concept $c$ can be denoted as follows:

$$\vec{F}_c = \{w_1, w_2, \ldots, w_k\}$$  (3.9)

where, $w_i$ is the weight of the corresponding word $a_i$ in a set of keywords \{$a_1, a_2, \ldots, a_k$\} extracted from the annotation of concept $c$.

The similarity between two concepts $c$ and $c'$ can be defined as follows:

$$\text{sim}(c, c') = \text{sim}(_\vec{F}_c, _\vec{F}'_{c'}) = \frac{\sum_{(k_i, k_i') \in K} (w_i \cdot w'_i)}{\sqrt{\sum_{i=1}^{n} (w_i)^2} \cdot \sqrt{\sum_{i=1}^{n} (w'_i)^2}}$$  (3.10)

where $k_i$ and $k'_i$ are keywords belonging to concepts $c$ and $c'$ respectively, $K = \{(k_i, k'_i)|\text{sim}(k_i, k'_i) = 1\}$

### 3.5.3 Structure Similarity

Structure-based similarity can be divided into two aspects the internal structure-based similarity and the external structure-based similarity.

The internal structure-based similarity [25] between two concepts is determined by the similarities between all pairs of their attributes. The similarity between two concepts $c$ and $c'$ can be computed as follows:

$$\text{Sim}(c, c') = \frac{\sum_{i=1}^{n} \text{Sim}(a'^c_i, a'^c_i)}{\max(|A'^c|, |A'^c'|)}$$  (3.11)

where

- $\text{Sim}(a'^c_i, a'^c_i)$ represents the similarity between attributes $a'^c_i \in A'^c$ and $a'^c_i \in A'^c$ belonging to two concepts $c$ and $c'$ respectively. Therefore, $\text{Sim}(a'^c_i, a'^c_i) = \text{Sim}(S_{a'^c_i}, S_{a'^c_i})$
• $|A^c|$ and $|A^c'|$ are the number of attributes belonging to these two concepts respectively.

• n is the number of pairs of attributes in different concepts with a degree of similarity greater than a given threshold.

The internal structure-based similarity here takes account into property restrictions for concepts; types, domains, and ranges for properties; and data values for individuals as [53, 54].

The external structure-based similarity considers the hierarchical relation that is-a links connect concepts that are already similar, therefore their neighbours may be also somehow similar. This intuition can be exploited in several different ways presented in [28].

3.5.4 Aggregated Similarity

Most of current ontology matching systems combine different similarity measures for enhancing ontology mapping. Aggregating different similarities is pervasive in ontology mapping systems that contain multiple individual matchers discussed in [69, 61]. Many strategies have been proposed to aggregate different similarities in the ontology mapping approaches such as Max, Weighted, Average and Sigmoid. Max strategy selects one maximum end of various similarities to be the representative of the final similarity. The average strategy returns the average similarity over all individual matchers. The Weighted strategy assigns relative weights to individual matchers. And, the SIGMOID strategy combines multiple results using a sigmoid function based on a threshold.

The Max strategy is too optimistic especially in case of contradicting similarities. The Average strategy considers the individual similarities
equally important and cannot distinguish differences between them. The Weighted strategy overcomes the drawbacks of the Average strategy by assigning relative weights to individual matchers. The SIGMOID strategy emphasizes high individual predicting values and deemphasizes low individual predicting values.

In most of the cases, the Weighted strategy needs to manually set aggregation weights using experience numbers and the Sigmoid strategy need to tentatively set center position and steepness factor in the sigmoid function. Machine learning based parameter optimization can be applied to overcome the manual parameter setting. However, learning based approaches need to consider additional information such as ground truth, which not exists in real world mapping task.

Note that previous kinds of similarities have been aggregated using an improved Harmony measure [69]. This measure identifies a number of real matches in each similarity matrix to assign a weight to the corresponding similarity measure for aggregation.
Chapter 4

Ontology Integration by Priorly Matchable Concepts Propagation

The aim of this chapter is to investigate a method to reduce the computational complexity and enhance accurate matching ontology by propagating Priorly Matchable Concepts (PMC). The key idea of the PMC approach is to analyze multiple contexts, including the role of natural categories in an ontological hierarchy, relations, and constraints among concepts to provide additional suggestions for priorly matchable concepts.

4.1 Priorly Matchable Concepts Notion

The priorly matchable concepts is based on two notions Concept Types and Concept Importance:

Concept Types, which was proposed in a previous research [28], is used to identify concepts playing a role of natural categories in ontology. The key idea of Concept Type, which is presented here, complements previous approaches, by analyzing the description of concepts, and by providing additional suggestions for possible matching concepts that if two ontologies need to integrate, two concepts crossing these ontologies in the same Concept Type should be priorly checked similarity.

Concept Importance is an importance measurement of a concept that shows how a central and richly described concept is in the taxonomy of the ontology. Therefore, it must take into account as the contributions from the concept’s attributes and all other concepts in the ontology.
Concept Importance is derived from the CARRank algorithm for concept ranking proposed by [38]. However, it is clearly different from the notion used in that algorithm. The importance measurement of the CARRank algorithm only takes into account the contribution from relations among concepts to improve the given initial importance of each concept. The key idea of Concept Importance, which is presented here, is to provide additional suggestions for possible matching concepts that two ontologies need to be integrated, two concepts belonging to different ontologies should be priorly checked for similarity, if their importance score similarity is greatest.

Priorly Matchable Concepts which is a collection of pairs of concepts across two different ontologies in the same concept type. These are arranged in descending order of concept importance similarity for pairs of concepts. Two concepts belonging to PMC with a greater similar score than other concepts should be priorly checked for similarity.

A novel idea of Priorly Matchable Concepts (PMC) propagation approach is that direct concept matching is driven between the same Concept Types and Concept Importance score, instead of exhaustive search among all concepts. This means that semantic commonalities between highly heterogeneous concepts can be achieved without taking an exhaustive search in taxonomies. Consequently, PMC propagation method supports content-based matching in a less complexity and name-base matching avoiding mismatching concepts.
4.2 Priorly Matchable Concepts Generation

Ontology reflects the creator’s own understanding of knowledge, like the relation of a literary work to its author. The best explanation of the phenomenon of human consciousness is William Jame’s famous stream of consciousness theory. He observed that human consciousness has a composite structure including substantive parts (thought or idea) and transitive parts (fringe or penumbra), and drifts from thought-to-thought. Thus, the independently developed ontologies of the same domain have heterogeneous characteristics and forms, e.g., languages, structures, etc. Therefore, ontology should be collaboratively developed by a large number of members. The goal of collaborative ontology building is to find a common ground, probing issues in ontology-building until everyone’s opinions are voiced and understood by the group [74]. To guide and evaluate the whole process of ontology building, [102] designed a criteria for ontology building such as clarity, coherence consistency, extensibility and minimal ontological commitment. If ontologies are designed using the above mentioned methods, the ontologies may effectively be communicated by agents, since they reduce mismatches among the entities of the ontologies. However, the computational complexity of matching among large ontologies still increases rapidly. It must be blind or exhaustive to check matchable among all pairs of concepts belonging to different ontologies. In this section, the novel approach to skip the unmatchable pairs of concepts by propagating PMC is presented. PMC is a collection of pairs of concepts belonging to two different ontologies in the same Concept Type, which are arranged in descending order according to the
Concept Importance score similarity of pair of concepts. The PMC guides on how to a priori check similarity between concepts. It is very useful to exponentially avoid checking the similarity of unmatchable concepts.

4.2.1 Concept Types

The Concept Types provide markers for concepts to distinguish among multiple perspective of the same concept. For example, the concept Computer Science in an ontology is described as Computer Science{About, hasStaff, hasCourse, hasStudent}, but in another ontology, the concept Computer Science is considered as an instance of the concept Fields. The meaning of the concepts named Computer Science in these two ontologies is different. In this case, designers can use Concept Type to mark these concepts with different categories. This way, it can be said that two concepts belonging to different ontologies in the same concept type should be a priori checked similarity. Here, concepts are classified into 4 concept types as follows:

Definition 4 (Defined Concept (DC)) The concept that has at least a set of attributes with necessary & sufficient conditions.

A set of attributes with necessary & sufficient conditions used to define a concept provides a unique value to each individual of the concept in the real world or possible world. If concept c is described using necessary conditions, then it can be said that if an individual is a member of concept c, then it must satisfy the conditions. It cannot be said that any (random) individual that satisfies these conditions must be a member of concept c. However, if concept c is now defined using necessary and sufficient conditions, then it can be said that if an individual is a member
of the concept $c$, then it must satisfy the conditions; further, it can now say that if any (random) individual satisfies these conditions, then it must be a member of concept $c$. The conditions are not only necessary for the membership of $c$ but also sufficient to determine that something satisfying these conditions is a member of $c$.

**Example 2** Consider concepts *Person* with *Fingerprint*, *PublishedBook* with *ISBN*. These concepts are defined concepts. A defined concept is also known as a rigid sort [44] that supplies a principle of identity for its individuals.

Notice that the top concepts are very often considered as defined concepts.

**Definition 5 (Partitive Concept (PC))** A partitive concept that is defined by specific concepts with constant attributes.

A constant attribute of a concept that provides a common attribute value for all individuals belonging to the concept.

**Example 3** The concept *FemalePerson* is defined by the concept *Person* with the constant attribute *Gender*. Its common value is *female*. This concept is a partitive concept.

**Definition 6 (Role Concept (RC))** A role concept that plays something, the concept must hold some object relations and can be divided into sub concepts.

**Example 4** We consider concept *Student* that enrolls in a department of an university, *Student* is divided into sub-concepts such as *BS*, *MS* and *PhD*. Another concept, *Employee*, that belongs to a department and can be divided into sub-concepts including *Teacher* and *Student* who has a part time job as a teacher. Accordingly, the concepts *Student* and *Employee* are role concepts.
Definition 7 (Primitive Concept (PvC)) A primitive concept that represents the possible stages of another concept and can not be divided into sub-concepts.

Example 5 We consider concepts PhDStudent and MscStudent as the possible stages of the concept Student; Woman and Girl are the possible stages of the concept FemalePerson; Grad-Course and U-Grad-Course are the possible stages of the concept Course. These concepts can not be divided into any sub-concept, so the concepts are primitive concepts.

Definition 8 (Concepts Types) For a given ontology O belonging to the real world (A,V), denotes five different sets of DC-, PC-, RC-, and PvC-concepts to be \(C_{DC}\), \(C_{PC}\), \(C_{RC}\), and \(C_{PvC}\) respectively.

1. \(C_{DC} \cup C_{PC} \cup C_{RC} \cup C_{PvC} = C\)
2. \(C_{DC} \cap C_{PC} \cap C_{RC} \cap C_{PvC} = \emptyset\)
3. The levels of concepts increase in the order of PvC, RC, PC and DC, respectively.

The Definitions shown that concepts belong to an ontology can be classified into five disjoint groups. DC, which includes top concepts in the ontological hierarchy, is highest level and PvC, which consists of leaf concepts in the ontological hierarchy, are lowest level.

Notice that Here, a simple algorithm for Concept Types classification in OWL ontology language is presented.

4.2.2 Concept Importance

Concept Types are important to distinguish among perspectives of the same concept in different ontologies. In ontology integration, two concepts that belong to different ontologies of the same concept type should
be a priori checked for similarity. However, if a concept type includes many concepts, then the number of pairs of concepts that should take a similarity check still rapidly increases. The Concept Importance measurement to solve this problem is proposed. It is an importance measurement of a concept as to how central and richly described the concept is in the taxonomy of the ontology. It must take into account the contributions from the concept’s attributes and all other concepts in the ontology. Assume that two ontologies need to be integrated, two concepts belonging

Figure 4.1: A simple algorithm for concept types classification
to different ontologies should be a priory checked for similarity if their importance score similarity is greater than other. That just consider checking the similarity between a concept belonging to an ontology and several concepts belonging to another ontology, respectively, depending on their importance similarity, where the concepts are the same concept type.

Cognitive support for ontology integration emphasizing Concept Importance has not yet been explored. However, Concept Importance have been considered to understand ontology in several early studies, e.g., [38]. They suggest that the importance measurement of a concept must take into account the contributions from all the other concepts in the ontology via characterization of four features of potentially important concepts and relations. These drive the drifting stream of consciousness:

- A concept is more important if there are more relations originating from it.
- A concept is more important if there is a relation originating from this concept to a more important one.
- A concept is more important if it has a higher relation weight with respect to any other concept.
- A relation weight is higher if it originates from a more important concept.

This work differs from the above approach, because the aim at this research is to design a method that reduces the computational complexity in ontology integration. First, it can be agreed that the importance measurement of a concept must take into account the contributions from all the
other concepts in the ontology based on all types of relations, including subsumption and non-subsumption. However, the Concept Importance should consider both the criteria drawn from cognitive science and the topological structure of the ontology; specifically, the importance measurement of a concept must take into account that (1) concepts play the role of natural categories in ontology and (2) how central and richly described the concepts are in the taxonomy of the ontology. In the first criterion (1), the importance concept should just consider the same concept type. In the second criterion (2), the importance concept should take into account the contribution from all its attributes and other related concepts. The Concept Importance measurement are formulated as follows:

\[
CI(c) = \frac{n_{isa}}{n} \sum_{c' \text{ isa } c} CI(c') + \frac{n_r}{n} \sum_{cRc'} IC(c') + \frac{n_c}{\max(n_{c'|c' \in C})} + \text{bias} \tag{4.1}
\]

where

Denotes \( n_{isa}, n_r, \) and \( n_{att} \) corresponding to the number of is-a relations, other binary relations, and attributes belonging to both ontologies that must be integrated, respectively; \( n = n_{isa} + n_r + n_{att} \); \( c' \text{ isa } c \) means that \( c' \) has a is-a relation with \( c \); \( cRc' \) meaning \( c \) has \( R \) relation with \( c' \); \( n_c \) is the number of attributes of the concept \( c \); and \( C \) is a set of all concepts that are the same Concept Type as the concept \( c \). One can regard bias as additional weight with constant input used to make \( CI(c) \) always different from zero. The \( CI(c) \) without bias will always be zero when the concept \( c \) has no attributes and is a leaf concept. It causes inefficient for calculating \( CI(c) \).
For simple understanding, it considers two small ontologies, as shown in the Figure 4.2. Relations study, teach and supervise are from domains Student, Teacher and Teacher to ranges Course, Course and Student respectively in both ontologies. It assumes that concepts have several datatype properties such as $O_1 : \text{Course}$ has datatype properties $c_1, c_2, c_3$ and $c_4$, as shown in Figure 4.2.

![Figure 4.2: Concept types and concept importance](image)

According to the definition of Concept Types and the algorithm shown in Figure 4.1, concepts of both ontologies can be classified into four concept types as follows: 

- $\{O_1 : \text{Person}, O_2 : \text{People}, O_1 : \text{Course}, O_2 : \text{Course}\}$,
- $\{O_2 : U - \text{GradCourse}, O_2 : \text{GradCourse}\}$,
- $\{O_1 : \text{Student}, O_1 : \text{Teacher}, O_2 : \text{Student}, O_2 : \text{Employee}\}$,
- $\{O_1 : \text{BS}, O_1 : \text{MS}, O_1 : \text{PhD}, O_2 : \text{BS}, O_2 : \text{MS}, O_2 : \text{PhD}\}$.

Calculates the concept importance score for each concept belonging to each concept type. The number of is-a relations, other relations
and attributes corresponding to $n_{isa}=17$, $n_{r}=6$, $n_{att}=23$. It also denotes bias to be zero. Following the Equation 4.1, The concept importance score for each concept is shown in Figure 2

$O_1: Person=0.42, O_2: People=0.46, O_1: Course=0.50, O_2: Course=0.47$, $O_2: U-GradCourse=0$, $O_2: GradCourse=0.50$, $O_1: Employee=0.47, O_1: Student=0.44, O_1: Teacher=0.50, O_2: Student=0.44, O_2: Employee=0.49$, $O_1: BS=0, O_1: MS=0, O_1: PhD=0, O_2: BS=0, O_2: MS=0, O_2: PhD=0$.

### 4.2.3 PMC Mapping Functions

Assume that two ontologies $O$ and $O'$ should be integrated. The instead of matching to all concepts by traversing taxonomies completely, each concept $c$ belongs to ontology $O$ tries to match only to a small number $k$ of concepts in ontology $O'$, which are the same Concept Type with concept $c$'s. The size of $k$ is determined by similarity between Concept Importance of the concept $c$ and the other concepts belonging to the ontology $O'$. This is an advantage over other existing mapping methods. This advantage comes from the following two postulates as follows:

1. There are no pair of concepts with different Concept Types across ontologies are semantic equivalent.

2. A concept belonging to an ontology only can match with a number of concepts belonging to another ontology, if and only if their Concept Importance core similarity are greater than a threshold. The threshold is given by user or measured based on Concept Importance-score similarity among concepts across these two ontologies.

We denote four different sets of $DC-, PC-$, and $RC-$ concepts to be $C_{DC}$, $C_{PC}$, and $C_{RC}$ respectively. Given two ontologies $O$ and $O'$. Denotes
\( f_{DC}, f_{PC}, \) and \( f_{RC} \) are PMC mapping functions that identify the corresponding concept \( c' \) of concept of \( c \in C_{DC}, C_{PC} \) or \( C_{RC} \) in \( C'_{DC}, C'_{PC} \) or \( C'_{RC} \), respectively, so that similarity of Concept Importance between the concepts \( c \) and \( c' \) are greater than a given threshold \( \varepsilon \). The matching functions are formalised as follows:

1. \( f_{DC} : c \in C_{DC} \Rightarrow c' \in C'_{DC}, |1 - (CI(c) - CI(c'))| \geq \varepsilon \)
2. \( f_{PC} : c \in C_{PC} \Rightarrow c' \in C'_{PC}, |1 - (CI(c) - CI(c'))| \geq \varepsilon \)
3. \( f_{RC} : c \in C_{RC} \Rightarrow c' \in C'_{RC}, |1 - (CI(c) - CI(c'))| \geq \varepsilon \)

where, \( c \) and \( c' \) are ontological concepts. \( IC(c) \) is Concept Importance function for the concept \( c \) (see Equation 4.1), however, \( IC(c) \) was normalised in a unite \([0..1]\).

### 4.3 PMC Propagation-based Matching Algorithm

The algorithm proposed here (called Anchor-Prior) using PMC propagation-based matching method. The term Anchor originates from the Anchor-PROMPT algorithm proposed in [82], which traverses paths between anchor points (entity pairs already identified as equal). Along these paths, new mapping candidates are suggested. Specifically, paths are traversed along hierarchies as well as along other relations. To take advantage of the idea of Anchor-PROMPT, an Anchor-Flood algorithm for ontology alignment was proposed in [68]. The central observation behind the Anchor concept is that if two concepts from two source ontologies are similar, then their neighboring concepts are often similar as well. These approaches require that initial anchors be provided such as
The Anchor-PROMPT method and must check similarities among all neighboring concepts across ontologies from the anchor to collect new matching concepts. The difference in the approach proposed here is that, instead of blindly checking similarities among neighboring concepts, the Anchor-Prior algorithm uses the PMC to provide additional suggestions for determining which pairs of neighboring concepts should be priorly checked for similarity, and instead of exhaustively finding anchors or pre-provided anchors, this approach generates initial anchors by priorly computing similarities between pairs of concepts belonging to DC, PC and RC, respectively. This algorithm uses a set of initial anchors that are derived from PMC. It outputs a set of matched pairs across different ontologies. Before proceeding to the details of the algorithm, terms frequently used in this paper must be defined.

4.3.1 Basic Notion

Similarly to [82, 68], the following definitions of Anchor and Neighbors are used here:

**Definition 9 (Anchor)** An anchor is defined as a pair of matched concepts in different ontologies O and O’. Formally, given two concepts c ∈ O and c’ ∈ O’, if c ≡ c’ meaning that c and c’ are matched, then X = (c, c’) is an anchor.

**Definition 10 (Neighbors)** A neighbor of concept c, which is a set of children, parents, grandchildren, and nephews of the concept c, can be defined as follows:

\[
\text{Neighbors}(c) = \exists_c \cup \{\cup_{c' \in \exists_c} \exists_c'\} \cup \exists_c \cup \{\cup_{c' \in \exists_c} \exists_c'\} \\
\cup \{\cup_{c'' \in \{\cup_{c' \in \exists_c} \exists_c'\} - \{c\}} \exists_c''\} \\
\] (4.2)

where, c is a concept, \(\exists_c = \{c' | c' \subseteq c\}\) and \(\exists_c = \{c' | c' \supseteq c\}\)

76
The *Anchor-Prior* algorithm generates *PMC* and *Prior* set that provide additional suggestions for determining which pairs of neighboring concepts should be priorly checked for similarity, instead of exhaustively finding among all combination of concepts across ontologies from an Anchor. Here, *PMC* and *Prior* set are defined as follows:

**Definition 11 (PMC)** A PMC is a collection of three categories DC, PC and RC in which pairs of concepts across ontologies in each category are arranged in descending order of the Concept Importance score similarity for each pair of concepts.

**Definition 12 (Prior)** Prior is a collection of combinations of pairs of concepts across the sets of Neighbor(c) and Neighbor(c') relative to a given Anchor(c, c'). It is sorted in descending order of the importance-score similarity of each pair based on PMC. The pairs which do not exist in PMC are appended to the end of Prior.

Prior are divided into five subsets as follows:

- Prior (children), Prior(Parent), Prior(Sibling), Prior(Nephew) are sets of pairs of concepts across Children, Parents, Siblings, Nephews sets of c and c', respectively.
- Prior (across) is a collection of combinations of pairs of unmatched concepts across the aforementioned Prior subsets.

### 4.3.2 Anchor-Prior Algorithm

**Simplified Illustration**

Before proceeding to the details of the algorithm *Anchor-Prior*, the algorithm is illustrated in a simplified form and with a comprehensive example. The example consists of two sample ontologies that to be integrated. The two ontologies #101 and #304 describing the domain of
bibliography are given in Figure 4.4, respectively. The ontologies are taken from the benchmark of Ontology Alignment Evaluation Initiative. The Anchor-Prior algorithm are divided into two following phrases initial anchors generation and neighbors exploring.

- Initial Anchors Generation, the instead of matching to all concepts by traversing taxonomies completely to find anchors between ontologies, at this step, it only considers on finding initial anchors from first three top levels of concept types DC, PC, and RC. These anchors, are very important matching concepts, are used to explore other commonalities between neighbors from the corresponding anchors. This is an advantage over other existing mapping methods such as [82, 68].

First of all, all concepts belonging to each ontology are classified into DC, PC, RC, and PvC Concept Types using algorithm as shown in Figure 4.1. The result of this step for two input ontologies #101 and #304 are presented in Table 4.1.

Notice that, the concept #304:Organization has no supper/sub concepts. Other concepts refer to it as an individual concept. Thus, it is classified to PvC concept type. However, the concept #101:Organization, that there are many sub-concepts and there is no any supper concepts, thus it is considered as DC concept type. Similar to #101:Chapter and #304:Chapter that are classified into different concept types. These pairs of concepts are considered as unmatchable concepts. The most of previous works usually take these mismatching concepts as semantically similar concepts, because their label and comment are similar.
Table 4.1: A classification of concepts belonging to ontologies #101 and #304

<table>
<thead>
<tr>
<th>Types</th>
<th>Ontology #101</th>
<th>Ontology #304</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>List <strong>Organization</strong></td>
<td>List <strong>Chapter</strong></td>
</tr>
<tr>
<td>PC</td>
<td>Informal</td>
<td>Informal Published Date</td>
</tr>
</tbody>
</table>
Next, concept importance-score for each concept is calculated using *Equation 4.1*. Finally, the PMC, is generated by using the PMC mapping, as shown in *Figure 4.3*.

<table>
<thead>
<tr>
<th>Defined Concepts (DC)</th>
<th>Partition Concepts (PC)</th>
<th>Role Concepts (RC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>101 204 CIS</td>
<td>101 204 CIS</td>
<td>101 204 CIS</td>
</tr>
<tr>
<td>List List 0.997</td>
<td>Informal Informal 0.828</td>
<td>... ...</td>
</tr>
<tr>
<td>... ... ...</td>
<td>... ... ...</td>
<td>5 Academic Academic 0.978</td>
</tr>
<tr>
<td>... ... ...</td>
<td>... ... ...</td>
<td>... ... ...</td>
</tr>
<tr>
<td>10 Part Part 0.861</td>
<td>11 Reference Entry 0.893</td>
<td>... ... ...</td>
</tr>
<tr>
<td>... ... ...</td>
<td>... ... ...</td>
<td>13 Book Book 0.690</td>
</tr>
</tbody>
</table>

*Figure 4.3: PMC for ontologies #101 and #304*

Eventually, initial anchors are generated by applying the similarity techniques for each pair of concepts in order of PMC. The initial anchors are very important matching concepts that facilitate to explore new matching concepts from their neighbors. The initial anchors are shown in the *Figure 4.4*.

Let us recall that the PMC mapping functions generate possible matching concepts between ontologies based on *Concept Types* and *Concept Importance*. If two concepts across ontologies belonging the same concept type and their similar importance score is greater a given threshold, the pair of concepts are considered as possible matching concepts. PMC contains sets of these pairs. If using PMC for matching process (assume that two ontologies O and O' should be integrated), each concept c belongs to ontology O tries to match only to a small number k of concepts in ontology O', which
are the same Concept Type with concept $c$'s. The size of $k$ is either determined by similarity between Concept Importance of the concept $c$ and the other concepts belonging to the ontology $O'$ or given by user.

- Neighbor Exploring starts off an anchor $(c, c')$, the Neighbor$(c)$ and Neighbor$(c')$ relative to the given anchor $(c, c')$ are generated. Each Neighbor set is segmented into different segmented sets (called Prior) of neighboring concepts. Each Prior set is a collection of
combinations of pairs of concepts across corresponding segments belonging to Neighbors (as presented in definition 10).

Assume that the anchor #101:Academic and #304:Academic are exploring. Then the algorithm generates the Neighbors as shown in Figure 4.5. Instead of blindly computing similarities among the concepts across the Neighbors as in [82], the algorithm Anchor-Prior developed here segments Neighbor into different segmented sets such as children, parents, grandchildren, siblings, nephews of the corresponding anchor and children of previously matched concepts (see Figure 4.5). Finally, Prior sets are generated. Similarity techniques are applied for each pair of concepts belonging to each Prior set to find new matching concepts.
Initial Anchors Generation Algorithm

The algorithm proposed here is novel in comparison with previous approaches [82, 68] in that it generates a set of matching concepts derived from DC, PC and RC are considered as the initial anchors (see Algorithm 1). The function Sim is an aggregated similarity function incorporating name, annotation, and structural similarities. A PMC is a collection of three categories DC, PC and RC in which pairs of concepts in each category are arranged in descending order of the Concept Importance score similarity for each pair of concepts. It is generated using PMC mapping functions.

```
input : Two ontologies O2, O1, and PMC
output: anchor set

1 foreach pair (c1, c2) ∈ PMC do
2    if (Sim (c1, c2) > threshold then
3        anchor ⇐ ∪ { (c1, c2) };
4    end
5 end

6 Return(anchor);
```

**Algorithm 1**: Initial anchors generation algorithm

Anchor-Prior

A simple form of the Anchor-Prior algorithm (see Algorithm 2) is presented here. It is novel from previous approaches [82, 68] in that it generates a set of initial anchors that are derived from PMC (at line 1). The initial anchors are considered as matched concepts at line 2. From line 3 to line 23, it explores new matching from anchors. A new matching may be
considered as a new anchor if the concepts belonging to the corresponding anchor are not leaf concepts (from line 12 to line 14). At line 5, the prior sets are generated from neighbors of an anchor. Each prior set is a collection of combinations of pairs of concepts across the corresponding sub-segments of neighbors, such as parents, children, grandchildren, siblings and nephews (see Figure 4.5). The pairs belonging to a prior set are sorted in descending order of the importance-score similarity of each pair based on PMC. The pairs which do not exist in PMC are appended to the end of Prior. This step differs from the previous approaches [82, 68] by using PMC for arranging possible concepts in a priority policy based on Concept Importance and Concept Types. The proposed algorithm outputs a set of matched pairs across different ontologies from the anchors at line 24 (see Algorithm 2 on the next page).
input : two ontologies $O_2$, $O_1$, and anchor
output: set of matched concepts

1 anchor is generated using Algorithm 1;
2 matchedSet $\leftarrow$ anchor;
3 foreach pair $(c_1, c_2) \in$ anchor do
4    Generates the neighbors $(c_1)$ and neighbors $(c_2)$ sets from the anchor $(c_1, c_2)$;
5    Generates the priorSet from the neighbors $(c_1)$ and neighbors $(c_2)$;
6   foreach prior $\in$ priorSet do
7      foreach pair $(c_1', c_2') \in$ prior do
8         if $(c_1', c_2') \notin$ matchedSet and $\text{Sim} (c_1', c_2') >$ threshold then
9            $\text{PvC} \leftarrow \text{PvC}/\{(c_1, c_2)\}$;
10           $\text{PvC} \leftarrow \text{PvC}/\{(c_i, c_j)\}$, i or j is equal to 1 or 2;
11           matchedSet $\leftarrow \cup \{(c_1', c_2')\}$;
12           if $c_1'$ or $c_2'$ are not leaf concepts then
13              anchor $\leftarrow \cup \{(c_1', c_2')\}$;
14         end
15      end
16     end
17 Remove prior from priorSet;
18 if priorSet $= \emptyset$ and anchor $= \emptyset$ and $\text{PvC} \neq \emptyset$ then
19     priorSet $\leftarrow \cup \{\text{PvC} \}$;
20     $\text{PvC} \leftarrow \emptyset$;
21 end
22 end
23 Return(matchedSet);

Algorithm 2: Anchor-Prior algorithm
Chapter 5

Experimental Result

The proposed algorithm Anchor-Prior (short name Aprior) is implemented using Java. It utilized the Jena Framework\(^1\) for parsing the ontologies, extracting the concepts. The input for the Anchor-Prior algorithm is only ontologies serialized using OWL.

We performed a comprehensive evaluation of the Anchor-Prior using third party datasets and other state-of-the-art systems in ontology matching. More specifically, the Anchor-Prior is evaluated in two different ways. Firstly, the ability of the Anchor-Prior is examined to serve as a general purpose ontology matching system, by comparing it with other systems on the Ontology Alignment Evaluation Initiative (OAEI) Benchmarks. Secondly, the Anchor-Prior is evaluated for the purpose of Linked Open Data (LOD) schema integration and compared it with other systems for ontology matching on LOD schema alignment.

5.1 Data Sets

Established in 2004 by leading researchers in the area of ontology matching, OAEI (Ontology Alignment Evaluation Initiative)\(^2\) is a coordinated international initiative to establish a consensus evaluation of available ontology integration methods. It provides Benchmarks\(^3\) data set with the aim at providing a large number of data sets for different ontology

\(^1\)http://www.openjena.org
\(^2\)http://oaei.ontologymatching.org
\(^3\)http://oaei.ontologymatching.org/2009/benchmarks/
integration situations to identify the strengths and weaknesses of each integration algorithm.

The domain of Benchmark data set is Bibliographic references. The data set is based on one particular ontology dedicated to the very narrow domain of bibliography and alternative ontologies of the same domain. The reference ontology is that of test #101. It contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. Participants manually discard various information from the reference ontology focused on the characterization of the behavior of algorithms in order to evaluate how algorithms behave when information is lacking rather than having them compete on real-life problems. They are organized in three groups:

- Data test 1 is consisting of simple tests (#101-104) such as comparing the reference ontology with itself, with another irrelevant ontology (the wine ontology used in the OWL primer) or the same ontology in its restriction to OWL-Lite;

- Data test 2 containing systematic tests (#201-266) obtained by discarding various information from the reference ontology focused on the characterization of the behavior of algorithms. The considered features were:
  - Name of entities that can be replaced by random strings, synonyms, name with different conventions, strings in another language than English;
  - Comments that can be suppressed or translated in another language;
- Specialization hierarchy that can be suppressed, expanded or flattened;
- Instances that can be suppressed;
- Properties that can be suppressed or having the restrictions on classes discarded;
- Data test 3 including four real-life ontologies of bibliographic references (#301-304) found on the web and left mostly untouched (there were added xmlns and xml:base attributes).

For data test 1, the edit distance-based similarity works very well for identifying similarities between ontologies. However, it is not effective in the cases of linguistic similarity, so the edit distance measure works very poor on data test 2. To overcome the limitation of the edit distance-based similarity, one solution is combination between the edit distance-based similarity and linguistic-based similarity (e.g., WordNet). But, in data test 2, there are many samples with ontological components replacing by random strings or another language than English, such as the samples #248-266, for which the structure-based similarity should be applied. Unfortunately, structural similarity contributes very little in the real-life data test 3 such as #301-304. For these reasons, the goal of using Benchmark series is to identify the areas in which the proposed matching algorithm is strong and weak.

Another real world data set is also used for evaluation. This data set contains schema-level mappings from two LOD ontologies to Proton [9] (an upper level ontology) created manually by human experts for a real world application called FactForge, with over 300 classes and 100 properties. These two LOD ontologies include:
• DBpedia\textsuperscript{4}: The RDF version of Wikipedia, created manually from Wikipedia article infoboxes. DBpedia consists of 259 classes ranging from general classes (e.g. Event) to domain specific ones (e.g. Protein).  

• Geonames\textsuperscript{5}: A geographic data set with over 6 million locations of interest, which are classified into 11 different classes.

Wikipedia defines Linked Open Data as a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF. In particular, links between LOD datasets are almost exclusively on the level of instances, and schema-level information is being ignored. The goal of using LOD set is to identify the behaviours of the proposed algorithm finding schema-level links between LOD datasets in the sense of ontology matching.

5.2 Evaluation Method

Here, the \textit{Precision} and \textit{Recall} measures are used to evaluate the proposed approach. The $N_{\text{total}}$ denotes the total number of pairs for matching concepts between the candidate ontologies by experts, $N_{\text{correct}}$ and $N_{\text{incorrect}}$ correspond to the number of correct pairs for matching concepts and the number of incorrect pairs for matching concepts sought by the proposed system, respectively.

\textit{Precision} is used to evaluate the ratio of incorrectly extracted rela-

\footnotesize{\textsuperscript{4}http://downloads.dbpedia.org/3.5.1/dbpedia 3.5.1.owl.bz2  
\textsuperscript{5}http://geonames.org}
Recall is used to evaluate the ratio of correct matching to the sought total by the system.

\[
\frac{N_{\text{correct}}}{N_{\text{correct}} + N_{\text{incorrect}}} \tag{5.1}
\]

Finally, the F-measure is calculated by combining precision and recall as follows:

\[
F = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{5.3}
\]

5.3 Evaluation Result

5.3.1 Similarity Techniques Evaluation

In the literature, the similarity between concepts across ontologies is determined by analyzing aspects such as textual annotation (id, label, comment and other annotations), external structure (parents and children), internal structure (property restrictions for classes; types, domains, and ranges for properties) and individual similarity. However, they are classified into three kinds of similarities consisting of edit distance based similarity, annotation similarity, and structure similarity. Each similarity measures the correspondence between two elements from a different perspective as mentioned in Chapter 3. A comparison of the similarity methods over all OAEI Benchmark data set are shown in Figure 5.1.

1. Edit distance works very well on the samples #101-104, #203, #208, #221-247 that have high similarity between the names of elements in ontologies. However, in the cases of linguistic similarity, it work very poor, e.g., #201 and #202. To overcome the limitation of edit
distance based similarity, one solution is combination between the edit distance-based similarity and linguistic-based similarity (e.g., WordNet). However, using WordNet-based similarity will cost much more time in finding synonymous relations between words, and thus decrease the efficiency of the whole approach as well as discussed on Chapter 3. Although, the WordNet-based similarity is integrated, it also works very poor in the cases #248-266. Because, the ontologies samples are scrambled labels, no comments, flattened hierarchy, no instance, and no property.

2. The structure similarity explores not only property restrictions for concepts, types, domains, and ranges for properties, but also parents, siblings and children relationships to determine for similarities. It works very well on the samples #248-266 which meaningful linguistic information has been removed or replaced by some randomly generated information. However, structure similarity contributes very little in real-life ontologies tests such as #301-304.

3. The annotation similarity takes advantage of both edit distance comparison and structure analysis.

5.3.2 Evaluation Based on Benchmark Track

In order to test the quality of matches generated using the Anchor-Prior in terms of two widely used matrices, Recall, and Precision, the proposed system is run on OAEI Benchmark data set and compared its performance with the best representative methods such as ASMOV [53, 54], RiMOM [98, 61], Lily [103] and Anchor-flood [68, 93] for Benchmark data set (2009).
Figure 5.1: A comparison of the similarity methods over all Benchmark data set
Figure 5.2 presents results on the matching track of the OAEI.

All aforementioned methods equally performs well for the sample data test 1. The data samples in the data test 1 are comprising of matching a single source ontology (#101) to the reference ontology with itself, with another totally different names, therefore only data test 2 and data test 3 are considered in comparison (see Figure 5.2). For the data test 2, all systems including Anchor-Prior show a drop in the performance with Recall measure. The reasons for this drop are as follows: (1) Some ontologies in data test 2 contain concepts from other than English. Thus systems which rely on lexico-syntactic tools obviously have difficulties with these ontologies. (2) Some of these ontologies consist of concepts with random names. So systems using the edit-distance based similarity are ineffective. (3) Some ontologies are flat taxonomy and without any properties where the matching should be not done on the basis of structure alone. Unlike the ontologies used in the data test 1 and the data test 2 which are obtained by discarding various information from the reference ontology #101, the data test 3 comprises of ontologies which have been created independently by other organizations and are used in the real world. The proposed method equally performs well for this data test in comparison with above mentioned methods. It also notices that the proposed algorithm get highest Precision over Benchmark data set than the other methods, since the proposed algorithm avoids computing similarities between mismatching concepts based on Concept Types and Concept Importance measures.

Here, a comparison of F-Measure with Anchor-Prior algorithm and the aforementioned methods is presented (see Figure 5.3). Results show
that the Anchor-Prior performed well among other methods in F-measures with three data sets as shown in Figure 5.3, the proposed method only shows lower scores compare with LILY in data test 2, but achieved better scores among others for data test 2. As mention earlier, the proposed method is suitable for matching well structured ontologies but the data test 2 does not fully provide that. Considering data test 3, which are the real-life ontologies the Anchor-Prior algorithm outperformed over the other algorithms.

According to Anchor idea, two versions of ontology matching algorithm have been implemented as follows: one using PMC, which is aforementioned Anchor-Prior, and the other without PMC such as Anchor-PROMPT (called Anchor). In both algorithms, the same similarity technique described earlier is used. From the results (Figure 5.4), it is apparent that the Anchor-Prior algorithm avoids computing a large number of unmatchable concepts, as shown by the number of computed
concept pairs. This means that the proposed algorithm has reduced the ontology integration search space while still guaranteeing highly accurate results, as shown by the values of Precision and Recall. Note that the proposed algorithm offers high values of Precision because the ontologies are segmented by classifying the concepts into disjoint concept types. Most pairs of concepts belong to different ontologies and different concept types; they are rarely similar (as discussed in [28]).

According to results shown in Figure 5.4, it is apparent that the Aprior algorithm avoided computing a large number of either unmatchable concepts or mismatching concepts. After eliminating these concepts, the number of computed concept pairs are very much less than the number of combination of concepts across ontologies as depicted in Figure 5.4, while the algorithm produced high accurate result (see Figure 5.3). This means that elimination of unnecessary concepts do not affect the integration accuracy. The reasons for this advantage are as follows: The Aprior
Figure 5.4: A comparison between the number of computed pairs of concepts and the number of all combination of concepts

segments ontologies by classifying concepts into disjoint concept types. Most pairs of concepts across different ontologies and different concept types, which are rarely similar, would be not considered. Addition, most concepts are semantic equivalent in DC, PC, and RC are direct matching based on Concept Importance-score, therefore unmatchable concepts could be not considered for similarity during matching process, and the proposed algorithm Aprior produced high accuracy (see Figure 5.3).

To avoid mismatching concepts and unmatchable concepts, a pre-process for concept classification and concept importance-score calculation is performed. Figure 5.5 shows the pre-process running time and matching running time for the proposed approach. However, the over-
all running time compared with another method depicted in Figure 5.6. Moreover, the results presented in Figure 5.6 show that the proposed algorithm’s running time is acquired better results compare with the representative method FOAM. After all, the proposed algorithm Aprior reduced the ontology integration space while still guaranteeing highly accurate results, as shown the values of Precision and Recall in Figure 5.2.

5.3.3 Evaluation Based on LOD Schema

We compared the performance of Anchor-Prior to existing solutions that performed well for LOD ontology alignment [51]. These solutions include:

- BLOOMS: This method is based on the idea of bootstrapping information already present on the LOD cloud. BLOOMS uses a rich knowledge Wikipedia to enhance semantical concepts before matching decision [51].
Figure 5.6: Running time comparison between Aprior and FOAM on Benchmark data set

- BLOOMS+: This method is extended from BLOOMS [52].
- S-Match: This method utilizes three matching algorithms basic, minimal, and structure preserving to establish mappings between the classes of two ontologies [40].
- AROMA: This method utilizes the association rule mining paradigm to discover equivalence and subclass relationships between the classes of two ontologies [10].

Figure 5.7 shows the results for all Anchor-Prior and the previous works BLOOMS, BLOOMS+, and S-Match. Table 5.1 and Table 5.2 show examples of correct and incorrect matching concepts generated by
Table 5.1: Sample of correct matches from LOD ontologies to PROTON generated by Aprior

<table>
<thead>
<tr>
<th>SportClub</th>
<th>SportsTeam</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle</td>
<td>MeanOfTransportation</td>
</tr>
<tr>
<td>ResearchOrganization</td>
<td>EducationalInstitution</td>
</tr>
<tr>
<td>WaterRegion</td>
<td>BodyOfWater</td>
</tr>
<tr>
<td>Country</td>
<td>Nation</td>
</tr>
</tbody>
</table>

Table 5.2: Sample of incorrect matches from LOD ontologies to PROTON generated by Aprior

<table>
<thead>
<tr>
<th>CommercialOrganization</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PeriodicalPublication</td>
<td>Work</td>
</tr>
<tr>
<td>AudioRecording</td>
<td>MusicalWork</td>
</tr>
</tbody>
</table>
Figure 5.7: LOD schema evaluation comparison between Aprior and the previous works BLOOMS, BLOOMS+, and S-Match.

Anchor-Prior from the two LOD ontologies to Proton.

Results show that the Anchor-Prior performed well among the other methods on both precision and recall, the proposed method only shows lower scores compare with BLOOMS+ on recall, but higher scores on precision. BLOOMS+ performed significantly better than all other methods on F-Measure. However, no method performed well on aligning Geonames with Proton. The only matching found by Anchor-Prior (and the other methods) is the class Country in Geonames is equivalent to the class Nation in Proton. The key reasons for the poor performance include: 1) Geonames has a small number of classes (and hence very limited contextual information) and 2) the names of the classes in Geonames are often vague and ambiguous (e.g. Code and Feature), which made it difficult to compute their similarity [52].
The result also shows that the proposed algorithm performed well on finding schema-level links between LOD datasets. The reasons for this significance are as follows: (1) Similar to BLOOMS+, Anchor-Prior uses a rich knowledge source i.e. WordNet to determine the similarity between the classes of two ontologies (see Equation 3.3 in Chapter 3); (2) The similarity method is applied to most matchable concepts during matching process. (3) Similar to BLOOMS+ and other methods, Anchor-Prior also uses contextual information from both WordNet and the ontologies being aligned to further support (or reject) an alignment.

Notice that the Anchor-Prior algorithm performs significantly for not only LOD datasets but also the Benchmark data set (see Figure 5.3 and Figure 5.8).

Figure 5.8: LOD schema evaluation comparison between Aprior and BLOOMS
5.4 Discussion

Ontology integration become an important problem in ontology engineering. The previous works on ontology integration have focused on similarity measurements between ontological entities, e.g., lexicons, instances, schemas and taxonomies, resulting in high computational costs due to the need to consider all possible pairs of concepts between given ontologies. The mismatching pairs of concepts have not yet become avoidable. In this paper, a novel approach is proposed for ontology integration in which Type Concepts and Concept Importance are proposed to guide on how to a priory check similarity between concepts. It is very useful to avoid checking similarity of unmatchable concepts and to reduce mismatching.

5.4.1 Concept Types

The Concept Types provide markers for concepts to distinguish among multiple perspective of the same concept. For example, the concept Computer Science in an ontology is described as Computer Science{About, hasStaff, hasCourse, hasStudent}, but in another ontology, the concept Computer Science is considered as an instance of the concept Fields. The meaning of the concepts named Computer Science in these two ontologies is different. In this case, designers can use Concept Types to mark these concepts with different categories. This way, it can be said that two concepts belonging to different ontologies in the same concept type should be a priory checked similarity. The four concept types such as Defined Concept, Partition Concept, Role Concept, and Primitive Concept are proposed in this
work. To identify concept types for concepts, concepts distribution in the ontological taxonomy are considered. The Figure 5.9 shows concept distribution for the ontology #101.

Figure 5.9: Concepts distribution in the ontological taxonomy

The *Primitive Concepts*, which are represented by single cycle (the cycle does not contain any other cycles), considered as leaf concepts. The top concepts are very often considered as *Defined Concepts* such as *Organization*. Notice that concepts *Conference, Journal* and *Person* are top concepts, however they are not defined concepts at all. A defined concept should be divided into sub-concepts, and there are no concepts (itself) referring to it. Other concepts are either *Partition Concepts* or *Role Concepts*. As discussing on Chapter 4, a partition concept has no object properties, and a role concept has at least one object property. The concept types of concepts belonging to the ontology #101 are shown in
Concept Types are important to distinguish among perspectives of the same concept in different ontologies. In ontology integration, two concepts that belong to different ontologies of the same concept type should be a priory checked for similarity, since the same type concepts have the same characteristic and structure.

5.4.2 Concept Importance

Concept Types are avoidable to calculate similarities between unmatchable concepts such as #101:Chapter and #304:Chapter or #101:Organization and #304:Organization. However, mismatching concepts still occur between domain ontologies, since such a ontologies have many similar concepts belonging to the same concept type. Intuitively, a concept plays a certainly important role in a particular ontology. The importance measurement of a concept must take into account the contributions from all the other concepts in the ontology via characterization of three features ISA relations, other relations (object properties), and attributes (data type properties). These drive the criteria for Concept Importance measurement:

1. A concept is more important if there are more relations originating from it.
2. A concept is more important if there is a relation originating from this concept to a more important one.
3. A concept is more important if it has more attributes.
4. A concept is more important if there are more ISA relations originating from it.
(5) A concept is more important if there is a ISA relation originating from this concept to a more important one.

Let us denote $n_{isa}, n_r$ and $n_{att}$ corresponding to the number of is-a relations, other binary relations, and attributes belonging to both ontologies that must be integrated, respectively; $n = n_{isa} + n_r + n_{att}; c' isa c$ means that $c'$ has a is-a relation with $c; cRc'$ meaning $c$ has $R$ relation with $c'; n_c$ is the number of attributes of the concept $c$; and $C$ is a set of all concepts that are the same Concept Type as the concept $c$.

The criterion (1) says that the concept importance of a concept is depended on the number of relations originating from it. The criterion (2) considers the importance of concepts to which the concept related. From these two criteria, the concept importance of a concept $c$ (denoted $CI(c)$) is determined by objective relations as follows:

$$CI(c) = \frac{n_r}{n} \sum_{c \in C} IC(c') \quad (5.4)$$

The criterion (3) means that the concept importance of a concept is counted by the number of the concept’s attributes.

$$CI(c)_{att} = \frac{n_{att}}{n} \frac{n_c}{\max(n_c | c' \in C)} \quad (5.5)$$

With the criterion (4) and (5), the concept importance of a concept is taken into account contribution from its sub-concepts.

$$CI(c)_{isa} = \frac{n_{isa}}{n} \sum_{c' \text{ isa } c} CI(c') \quad (5.6)$$

From aforementioned analysis, the Equation 4.1 in Chapter 4 is generated for Concept Importance measure.
Chapter 6

Conclusion

In this Chapter, I conclude this thesis with a discussion of four topics. Firstly, the main contributions of this thesis are summarized. Secondly, a list of advantages and limitations of ontology integration by propagating a context in priorly matchable concepts are given. Thirdly, a brief history of my research progress is described according to the list of publications. Finally, a number of directions for future work are presented.

6.1 Summary of Contributions

First of all, the main goal of this research is to deal with wide-scale semantic heterogeneity in ontology matching. I focus on two main issues. The first issue is to avoid computing similarities between mismatching concepts and unmatchable concepts across ontologies. The second issue is to reduce complexity, concerning wide-scale semantic heterogeneity in content-based matching. To accomplish this, instead of computing similarities between all combinations of concepts across the ontologies, matchable concepts are suggested in a priority policy during matching process.

I describe a summary of my contributions according to these issues as well as the objectives already defined in Section 1.3. In this thesis, finding semantic commonalities between two heterogeneous ontologies is approached. It focuses on a matching method between classified concepts using the most similar concept importance-score which is used
to suggest matchable concepts between two corresponding ontologies.

The whole approach can be partitioned into two phases: (1) the priorly matchable concepts (PMC) generation and (2) the PMC propagation-base matching method. The former phase consists of modeling and implementing PMC. The later phase is for a matching method called Anchor-Prior, a particularised PMC and an experiment. Below are the contributions of each phase in details:

1. A novel approach of Priorly Matchable Concepts propagation are proposed, regarding that direct concept matching is driven between the same Concept Types and Concept Importance score. This means that semantic commonalities between highly heterogeneous concepts can be achieved without taking an exhaustive search in taxonomies. Consequently, PMC propagation method supports content-based matching in a less complexity and name-base matching avoiding mismatching concepts, in which priorly matchable concepts are determined by two measurements Concept Types and Concept Importance as follows:

   - All concepts belonging to an ontology are classified into disjoint categories (called Concept Types) by analysing taxonomy dependency and types of properties. The innovation behind Concept Types is to supply an additional suggestion for identifying possible matching concepts, regarding that if two concepts are semantically equivalent, then they must be classified within the same concept type.

   - Concept Importance is a measure of the importance of a concept that shows how centrally located and richly described
a concept is in the taxonomy of an ontology. Therefore, the measurement must take into account as contributions from the concept’s attributes and from all other concepts in the ontology through their relations. The Concept Importance is to supply an identifiable link between two heterogeneous descriptions of a concept, regarding that if two ontologies need to be integrated, two concepts in different ontologies should be priorly computed for similarity if their importance-score are most similar than other pairs.

2. PMC propagation -based matching method is an effective matching algorithm (called Anchor-Prior). It is an effective approach for improving of Anchor algorithms such as Anchor-PROMPT which traverses paths between anchor points (entity pairs already identified as equivalent). Along hierarchies or other relations, new matching candidates are suggested. The importance in Anchor algorithms is both of measures how to identify anchors and how to priorly matching among neighbors of an anchor. Previous works on Anchor idea have not yet explored these problems. Anchors should be priorly collected from highest concept type DC to lowest concept type PvC. However, initial anchors should only consider in first three concept types including DC, PC and RC. Then starting from an Anchor to work towards a collection of matched pairs among its neighboring concepts by computing similarities between the priorly collected concepts across the ontologies starting from the anchor.

3. The method is implemented in Java for matching between OWL
ontologies by utilizing Jena OWL API. Experimental results shows that our algorithm perform significantly in terms of accuracy and running compare with some best know methods from OAEI evaluations. We have also tested our system with Linked Open Data set and acquired better results compare with the existing methods.

6.2 Advantages and Limitations

The advantages of the priorly matchable concepts - based matching ontology over other mapping methods are discussed as follows:

1. Direct concept matching is initiated between the same Concept Types and between most similar Concept Importance-score, instead of a blind or exhaustive matching among all concepts across ontologies. It has been experimented that the approach is effective both in terms of performance and accuracy, by avoiding an exponential increase in computing similarities between unmatchable concepts across ontologies, and in term of accuracy as shown by the high values of Precision and Recall in comparison with the best representative methods of Benchmark data set. Note that the proposed algorithm achieves high values of Precision because it segments ontologies by classifying concepts into disjoint concept types. Most pairs of concepts belong to different ontologies and different concept types; they are rarely similar. Addition, most concepts are semantic equivalent in DC, PC, and RC are direct matching based on Concept Importance-score, therefore unmatchable concepts could be not considered for similarity during matching process.
2. In comparison with previous researches of Anchor-based matching such as Anchor-PROMPT and Anchor-Prior, PMC propagation-based matching considers how to generate initial anchors that fulfil the previous Anchor-based matching methods. The initial anchors are derived from DC, PC, and RC to segment matchable concepts belonging to PcC for priorly matchable concepts propagation.

Although the priorly matchable concepts-based matching has an advantage in directly finding commonalities between ontologies, the method still will be the cost of integrating either different domain ontologies and candidate ontologies are flat structure or only consisting of commonalities in bottom of taxonomy of the ontologies.

6.3 A Brief History of Research Progress

In this section, a brief history of my research and progress according to the listed publications is described. It started with a study of ontological analysis and similarity measures for ontology integration. I first worked out the presentation of similarity measure using WordNet and lexico-syntactic pattern (see [26, 27]). Then, I have focused on methods for ontology integration and started to a hybrid method for integrating multiple ontologies (see [24, 28]). I have learnt that Ontology integration is a complex task, since ontologies have various characteristics and forms, such as languages and domains, structures of ontologies may differ from each other. Therefore, I move to complexity analysis of ontology integration methodologies with a study case (see [21]) in which I found out ideas of Concept Types and Concept Importance measurements to enhance current ontology integration methods (see [29, 32]). Ultimately, I
successfully proposed the PMC-based matching ontology (see [18, 25]).
Then, I have move to fuzzy ontology integration (see [74, 31, 30, 25]).
Addition, I proposed methodologies for ontology building (see [74]) and various applications of ontologies (see [75, 74, 72]).

6.4 Future Directions

I would like to discuss on some major future direction issues on ontology integration.

1. Currently, this research work focus only on ontology matching with finding equivalent relations between concepts and properties across ontologies. Finding more complex relations such as subsumption relations (relates more specific to more general concepts in conceptual taxonomies) and disjoint relation will be more challenging. Future work in this direction could integrate an external repository to analyze the hierarchical relationship between classes, which are not explicitly expressed in the two ontologies.

2. Currently, it uses only three similarity measures such as edit distance, annotation-based similarity, structure-based similarity. However, these similarity measures are lack of semantic expression of concepts. In future work, I will focus on a semantic similarity measure that should cover following challenges:

   (a) Multiple forms of the same concept means ontologies define the same concept in different ways. For example, concept Person in one ontology may be defined by attributes: Name, Age, Address, Sex, Job, while in another ontology it is defined by
attributes:  Id, Name, Address, Date_of_birth, Taxpayer_identification_number, Occupation.

(b) Overlapping but different concepts means ontologies define different concepts in the same name and structure. As an example, we consider two concepts with the same name and structure which both contain knowledge about student. First the concept structures its knowledge of female student, whereas the second structures its knowledge of male student.

(c) The same concept but different names means that ontologies define the same concept in different names. For example, while defining concept Person, one ontology defines the concept name of Individual but another defines the concept name of Homo.

(d) The same name but different concepts means that ontologies define different concepts in the same name. For example, while using the same name of Master Course, one ontology considers the concept of subjects for a master student, while another ontology considers the concept of a master student.

(e) The multiple concepts of the same form means that ontologies define different concepts in the same structure.
REFERENCES


118


Local Context for Semantic Matching. In Workshop on Ontologies
and Information Sharing at the Seventeenth International Joint Con-
fERENCE on Artificial Intelligence (IJCAI-2001), Seattle, WA. pp. 242 -
258.

Tools For Ontology Merging And Mapping, In International Journal
of Human-Computer Studies, 59 (2003), 983-1024.

Tools For Ontology Merging And Mapping; In International Journal
of Human-Computer Studies, 59 (2003), 983-1024.

[85] Obrst, L., Wray, R.E., Liu, H.: Ontological engineering for B2B E-


[88] Phytila, Chris. An Analysis of the SUMO and Description in Unified

[89] Pinto, H.S., and Martins, J.P.: A Methodology for Ontology In-
tegration, In: Proceedings of the First International Conference on


Acknowledgement

Looking back, I am surprised and at the same time very grateful for all I have received throughout these years. It has certainly shaped me as a person and has led me where I am now. All these years of M.Sc. and Ph.D. studies are full of such gifts.

First of all, I would like to express my sincere gratitude to Professor Jo, Geun Sik from the School of Computer and Information Engineering, Inha University, Korea, to be my supervisor for my integrated M.Sc. and Ph.D. studies. He has provided me with many helpful suggestions and important advices, and has supported me opportunities and means for the scientific activities during the course of this work.

I also wish to express my deepest gratefulness to Professor Nguyen, Ngoc Thanh from the Institute of Informatics, Wroclaw University of Technology, Poland, who personally selected me as a candidate for a Ph.D. to receive a scholarship from Inha University, Korea, his valuable suggestions, constructive advices and constant encouragement during my study of this work are really appreciated.

I would like to express my sincere thanks to all Professors and Teachers in Quang Binh University, Hue University, Roon High School, and Canh Duong Elementary School, Vietnam, who taught me principles of life and valuable knowledge.

I would like to express my sincere thanks to all Professors in the School of Computer and Information Engineering, Inha University, Korea, for their helps and valuable lectures.

I would like to express my special thanks to all Professors in the committee of my Ph.D. dissertation defense Professor Kang, Sanggil,
Professor Jo, Geun Sik, Professor Lim, Joon Sik, Professor Lee, Jae-Dong, and Professor Lee, Sang-Chul for their helps and valuable advices.

I would like to thank all my fellow lab members in Intelligence E-commerce Systems Lab for the stimulating discussions, for the sleepless nights that we were working together before deadlines, for help, support, interest, valuable hints and for all the fun we have had during these four years. Also I would like to thank my friends in Inha University for their helps and encouragement.

My special gratefulness goes to my parents, Duong Hoai Chau and Nguyen Thi Coi, who always kept me away from family responsibilities and encouraged me to concentrate on my study.

My special appreciation goes to my family in law, mother Ho Thi Mau, sisters’ family and uncle Nguyen Duc Viet, who has supported me from difficulty beginning days and taken care my wife and son.

I offer my regards and blessings to all of my neighbors and those who supported me in any respect during the completion of this work.

Last but not least, I would like to express special thanks to my wife Nguyen Thi To Nga whose patient love. She has been taken good care of our son Duong Gia Bao and family, and supported mentally during the course of this work that enabled me to concentrate on completing this dissertation.

I have finished with a final silence of gratitude for my life.