ABSTRACT

ONTOGRAPHY ALIGNMENT TECHNIQUES FOR LINKED OPEN DATA ONTOLOGIES

by Chen Gu

Ontology alignment (OA) addresses the Semantic Web challenge to enable information interoperability between related but heterogeneous ontologies. Traditional OA systems have focused on aligning well defined and structured ontologies from the same or closely related domains and producing equivalence mappings between concepts in the source and target ontologies. Linked Open Data (LOD) ontologies, however, present some different characteristics from standard ontologies. For example, equivalence relations are limited among LOD concepts; thus OA systems for LOD ontology alignment should be able to produce subclass and superclass mappings between the source and target. This thesis overviews the current research on aligning LOD ontologies. An important research aspect is the use of background knowledge in the alignment process. Two current OA systems are modified to perform alignment of LOD ontologies. For each modified OA system, experiments have been performed using a set of LOD reference alignments to evaluate their alignment results using standard OA performance measures.
ONTIOLOGY ALIGNMENT TECHNIQUES FOR LINKED OPEN DATA ONTOLOGIES

A Thesis

Submitted to the

Faculty of Miami University

in partial fulfillment of

the requirements for the degree of

Master of Computer Science

Department of Computer Science and Software Engineering

by

Chen Gu

Miami University

Oxford, Ohio

2013

Advisor__________________________________________

Valerie Cross, PhD.

Reader____________________________________________

Eric Bachmann, PhD.

Reader____________________________________________

Michael Zmuda, PhD.
# TABLE OF CONTENT

1. Introduction ........................................................................................................................................ 1  
2. Linked Open Data .................................................................................................................................. 2  
3. Ontology Alignment ............................................................................................................................. 3  
4. Previous Research to Align LOD Ontologies .......................................................................................... 7  
  4.1 BLOOMS and BLOOMS+ .................................................................................................................. 7  
  4.2 WikiMatch ......................................................................................................................................... 9  
  4.3 AgreementMaker ............................................................................................................................... 10  
  4.4 Previous Experiments to Align LOD Ontologies .............................................................................. 11  
5. WikiMatch for Aligning LOD Ontologies ............................................................................................... 12  
  5.1 Initial WikiMatch Experiments ........................................................................................................ 12  
  5.2 Adding Subclass/Superclass Mapping Capability to WikiMatch ....................................................... 14  
  5.3 Experiments with Modified WikiMatch .......................................................................................... 16  
6. LogMap for Aligning LOD Ontologies .................................................................................................. 18  
  6.1 Background for Selection of LogMap .............................................................................................. 18  
  6.2 LogMap Overview ............................................................................................................................ 19  
  6.3 WikiNet ............................................................................................................................................ 21  
  6.4 LogMap + WikiNet Algorithm ........................................................................................................ 23  
  6.5 LogMap + Inferencing on Equivalence Mappings ........................................................................... 25  
  6.6 LogMap + Inferencing on Equivalence Mappings + WikiNet Algorithm ......................................... 34  
7. Conclusions and Future Research Directions ....................................................................................... 39  
Reference .................................................................................................................................................. 40
LIST OF TABLES

Table 4.1  BLOOMS and AgreementMaker (AM) Results on LOD Reference Alignment.......................... 12
Table 5.1 Original WikiMatch on OAEI 2012 and 2013 Conference Track Test Case............................... 13
Table 5.2 Original WikiMatch on Five Pairs from the LOD Reference Alignments ............................... 14
Table 5.3 Modified WikiMatch on Five Pairs from the LOD Reference Alignments .............................. 17
Table 6.1 Original LogMap on Five Pairs from the LOD Reference Alignments .................................. 21
Table 6.2 LogMap + WikiNet Algorithm for LOD Reference Alignment ................................................. 25
Table 6.3 LogMap + Level 1 Inferencing Results for LOD Reference Alignments ................................. 29
Table 6.4 LogMap + Level 2 Inferencing Results for LOD Reference Alignments ............................... 34
Table 6.5 LogMap + Level 2 Inferencing + WikiNet Algorithm .............................................................. 36
Table 6.5a Two level Inferencing Results for Comparison ................................................................. 36
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>General Architecture for OA systems (Erhig 2006)</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>WikiMatch Data Levels (Hertling and Paulheim 2012)</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>LogMap nutshell (Jiménez-Ruiz and Grau 2011)</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>Relations and some instances induced for MIXED MARTIAL ARTS TELEVISION PROGRAMS (Nastase and Strube 2013)</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>One-level inferencing</td>
<td>27</td>
</tr>
<tr>
<td>6</td>
<td>AGENT to group of person example</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>Two-level inferencing only on Source and Target Concepts</td>
<td>31</td>
</tr>
<tr>
<td>8</td>
<td>Two-level inferencing not only on Source and Target Concepts</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>person category tree and country category tree</td>
<td>38</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I would like to give my sincere thanks to Prof. Valerie Cross for her academic advice, constant encouragement and valuable help on both this thesis and my study at Miami University. To my thesis committee members: Prof. Eric Bachmann and Prof. Michael Zmuda, thank you very much for your time in reviewing my thesis and for your comments and suggestions.

During the implementation phase of this thesis work, Dr. Ernesto Jimenez-Ruiz, a research assistant in the Information Systems Group, Computer Science Department at the University of Oxford had provided important technical advice on LogMap, an ontology alignment systems used in this thesis research, and support when problems arose in adding new algorithms within LogMap. I hereby thank him for serving as an amazing consultant.

Finally I would like to give my special thanks to my family.
1. Introduction

Nowadays, the method of sharing documents and data has changed greatly by the rapidly growing World Wide Web (WWW). The WWW allows us to readily access and publish documents and data and these user capabilities make the World Wide Web extensible (Bizer et al. 2012). Though the WWW has benefited people in diverse areas, with the rapid development of hardware storage and information explosion, the proliferation of data has been a significant problem to sustaining the efficiency and usability of the WWW. The traditional method treats data as raw dumps in formats such as CSV or XML, or marked up as HTML tables. Such method cannot maintain data’s structure and semantics. However, in recently years, a novel concept, the Semantic Web, promotes common data formats on the World Wide Web. Its goal is converting the current web dominated by unstructured and semi-structured documents into a "web of data". As Tim Berners-Lee describes, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries." (Berners-Lee et al. 2001). However, since different standards and different creators exist, how to enable information interoperability between ontologies is a significant challenge for the Semantic Web. This thesis research addresses this challenge by using knowledge resources on the Semantic Web to perform ontology alignment (OA) between linked open data (LOD) ontologies.

The linkage between instance data for Linked Open Data (LOD) datasets is becoming well established (Voltz et al. 2009); however, the challenge is to improve the alignment between the LOD ontologies. Research in ontology alignment (OA) (Euzenat et al. 2011) has primarily focused on taking as input two well defined and structured ontologies from the same or closely related domain and producing a high-quality alignment between the source and target ontologies. For the most part, the type of mappings found in the alignment is equivalence between concepts in the two input ontologies. These features used by traditional OA systems are not as suitable for aligning LOD ontologies; for example, equivalence mappings are limited among LOD ontology concepts. The scarcity of such equivalence mappings necessitates that OA systems for LOD ontology alignment also find subclass and superclass mappings.

This thesis research makes several contributions to ontology alignment research.
1) Current techniques used in recent OA systems to align LOD ontologies are investigated.
2) The use of Wikipedia as background knowledge for ontology alignment is examined in the only two OA systems which have experimented with its use.
3) Several algorithms have been implemented in two different current OA systems to add the capability of aligning LOD ontologies.
4) Evaluations of these implementations have been performed using an LOD reference alignment previously used by two current OA systems and the standard OA performance measures.

These contributions help to advance ontology alignment research in two areas, the use of background knowledge in the OA process and adapting traditional OA systems to handle LOD ontologies.

The remainder of the thesis is organized as follows. Section 2 describes what linked open data is and how it is being used is. An overview of the ontology alignment process is presented and two leading ontology OA systems which have been investigated as part of this thesis research are described in section 3. Current research in the alignment of LOD ontologies using background knowledge sources is reviewed in section 4. Based on this review, sections 5 and 6 describe the work done to modify two existing OA systems to perform alignment on LOD ontologies. Each of these sections presents the results of experiments using these modified systems. Conclusions and directions for future research are presented in section 7.

2. Linked Open Data

Linked data is a term for describing a method for creating typed links between data from different sources. Due to the ability to link data, the web has improved from a global information space of linked documents to a global space where both documents and data are linked. By using linked data, user can traverse from data in one data source to data in another source. Moreover, sophisticated querying capabilities can be used over the linked data. For achieving such work, linked data should be published on the web and in such way that software can understand the meaning of data.

Berners Lee has outlined four principles for publishing such data (Bizer et al. 2012):
1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL, N3)
4. Include links to other URIs, so that they can discover more things.

An uniform resource identifier (URI) is a string of characters for identify a resource. It enables interaction with representations of the resource over World Wide Web by using specific protocols. Resource Description Framework (RDF) is a specification published by W3C which is used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax formats. Resource Description Framework Schema (RDFS) is an extension of RDF to define a set of classes with properties that provide the basic constructs needed to describe ontologies also referred to as RDFS vocabularies.

SPARQL Protocol and RDF Query Language (SPARQL) is an RDF query language that enables querying data stored in RDF format. The query format typically is in triple notation patterns (subject, verb, object) and as in other database query languages allows for disjunction and conjunction of queries. Notation-3(N3) is a shorthand non-XML serialization for representing RDF descriptions and follows the (subject, verb, object) pattern. As a logic language, it is a superset of RDF since it extends the RDF data model with formulae, variables, logical implication, and functional predicates.

Linked open data is Linked Data which is released under an open license, which permits free reuse. For example Linking Open Government Data (LOGD) enables the sharing and reuse of data (DiFranzo 2010). LOGD translates government-related datasets into RDF, linking them to the web of data and providing application for searching via SPARQL. All linked data, however, does not necessarily have to be open.

3. Ontology Alignment

Much of the research in ontology alignment has focused on finding equivalence mappings between classes between two ontologies. Some OA systems also provide equivalence mappings
between properties within the ontologies being aligned. The Ontology Alignment Evaluation Initiative (Euzenat et al. 2011) has also concentrated on evaluating OA systems based on equivalence mappings between classes. Aligning LOD ontologies, however, also requires producing subset and superset mappings between source classes and target classes. The following first provides an overview of the OA process and then describes the OAEI competition.

3.1 OA Process

Ontology alignment is a process that produces a set of correspondences between entities (concepts and properties) in two or more ontologies. There are many current OA systems, but most of them follow a general architecture as presented in figure 1.

Figure 1 General Architecture for OA systems (Erhig 2006)

The input is two or more ontologies which need to be aligned.

1. Feature engineering requires extracting information that describes specific entities from the overall ontology definition for similarity comparison in later steps. For example, an alignment system may extract concepts’ labels and comments for later comparison.
2. Search Step Selection must determine which entities pairs to be compared.
3. Similarity Computation calculates the similarity between two entities which are determined as candidates to be aligned.
4. Similarity Aggregation takes the individual similarity values for one candidate pair calculated in the similarity computation step using several different matching algorithms and combines them into a single aggregated similarity value.
5. The Interpretation step uses individual or aggregated similarity values to derive alignments between entities. Finally, a proposed alignment (or not) for the selected entity pairs is returned.
6. Iteration occurs since the similarity of one entity pair may affect the similarity of other entity
pairs. During an iteration, the similarity of a candidate alignment is recalculated based on the similarities of neighboring entity pairs.

Note that in a subsequent iteration, one or several of steps 1 to 5 may be skipped because all features might already be available in the appropriate format or because similarity computation might only be required in the first round iteration.

Much research in OA has been done in the past decade and an official competition, the Ontology Alignment Evaluation Initiative (OAEI), was started in 2004 to compare the performance of the various existing OA systems (Euzenat et al. 2011a). The following section briefly describes this competition.

### 3.2 Ontology Alignment Evaluation Initiative

The OAEI is a coordinated international initiative to evaluate the performance of various OA systems by using a standard set of test cases. Its objectives are assessing strengths and weaknesses of alignment/matching systems, comparing performance of techniques and increasing communication among algorithm developers (Euzenat et al. 2011). Based on these objectives, the OAEI can help improve the research on ontology alignment/matching through the controlled experimental evaluation of these systems. The means to achieve these goals are organization of a yearly evaluation event and publication of the tests and results of the event for further analysis.

There are various test tracks in each yearly OAEI competition, such as the anatomy track which is about matching the Adult Mouse Anatomy (2744 classes) and the NCI Thesaurus (3304 classes) describing the human anatomy. The Benchmark track has several different ontologies it uses as the basis to randomly generate versions of that ontology by randomly altering various pieces of information in the ontology such as concept labels and comments or its structure. A bibliographic ontology has been the primary ontology used for this track. The benchmark track is for identifying the areas in which each alignment algorithm is strong or weak.

In the OAEI competition, three measures have been used for evaluating the performance of OA
 systems: precision, recall and the f-measure. For calculating these measures, a reference alignment which is regarded as the correct alignment result is required. This alignment is also referred to as the gold standard alignment.

Given the reference alignment R and an alignment A produced by an OA system, precision is the fraction of matched entities in the alignment A that are in the reference alignment R, i.e., are correct:

\[ P(A, R) = \frac{|R \cap A|}{|A|} \]

Recall is the fraction of matches in the reference alignment R that are in the produced alignment A:

\[ R(A, R) = \frac{|R \cap A|}{|R|} \]

F-measure is a measure that combines precision and recall using a parameter \( \beta \):

\[ F_\beta = (1 + \beta^2) \cdot \frac{P(A, R) \cdot R(A, R)}{\beta^2 P(A, R) + R(A, R)} \]

The weighting factor \( \beta \) allows emphasizing either recall or precision. When \( \beta \) is 1, \( F_1 \) gives equal importance to each and the harmonic mean of the two is calculated. In recent OAEI competition, two additional kinds of F-measure have been used to emphasize either precision or recall. \( F_{0.5} \) weights precision higher than recall while \( F_2 \) weights recall higher than precision.

In 2010 OAEI competition for the anatomy track (Euzenat 2010), AgreementMaker(Cruz 2010) had one of the best overall performances. Because of these results and also because of the interest of AgreementMaker’s developers, it was used in Xueheng Hu’s thesis research (Hu 2011). AgreementMaker at that time was not open source, but the source was provided to Dr. Cross by the Advances in Information Systems Laboratory (ADVIS) at the University of Illinois, Chicago to be used for masters’ thesis research projects. AgreementMaker had been under consideration for use in this thesis research. However, due to issues with getting LOD reference alignments (Jain et al 2010) also used by AgreementMaker and a disagreement on crediting its previous usage, its license for use by Dr. Cross was revoked this past summer. At that point, it became necessary to find another OA system for purposes of this thesis research. LogMap has successfully participated in the last several years of the OAEI event and had very
competitive results as compared to the other OA systems. It was the leading OA system in the 2012 competition (Euzenat et al. 2012). The one important factor that LogMap excels in compared to all the other OA systems is its very low runtimes. Because LogMap is available through open source, participated in OAEI 2013 and the LogMap developers provided support, a major component of this thesis research involves the modification of LogMap to produce alignments between LOD ontologies. A description of LogMap is provided in section 6.1.

4. Previous Research to Align LOD Ontologies

Recently researchers have begun to explore the use of Wikipedia as a background knowledge source in the OA process. Wikipedia (Rosenzweig 2006) is one of the largest knowledge compilations and covers very diverse domains. Although several researchers have recommended Wikipedia for use in the OA process, only two OA systems have actually implemented its use. Another OA system uses WordNet (Miller et al. 1990) as its background knowledge. These three systems are described in the following sections since some of their techniques are implemented and modified as part of this thesis research.

4.1. BLOOMS and BLOOMS+

Bootstrapping-based Linked Open Data Ontology Matching (BLOOMS) system (Jain et al. 2010) and BLOOMS+ (Jain et al. 2011) are OA systems for automatically finding correspondences between LOD ontologies. Both uses Wikipedia as a knowledge source to construct a set of category hierarchy trees called a forest for each concept in the source and each concept in the target ontology.

Both BLOOMS and BLOOMS+ use the same procedure to build the category trees. For each class C from both source and target ontology, BLOOMS+ tokenizes the name of C and removes stop words from the name and then uses the resulting terms as a search string to retrieve relevant Wikipedia article pages using Wikipedia search web service API. Multiple pages may be returned by the Wikipedia search. BLOOMS+ considers each page as a root and then constructs a category hierarchy tree by following steps:

1. Set the article page as root node
2. Find all Wikipedia categories that this node belongs to as immediate children.
3. Generate every subsequent level of this tree by doing step 3 recursively for each child node.

Through experimentation, the BLOOMS developers set the limit to 4 for the category tree depth since they found that creating the category tree below that depth typically includes very general categories which are not useful in the alignment process. All the trees generated for class C are grouped together and referred to as the forest of C.

BLOOMS uses a simplistic inclusion measure calculated as the fraction of the total number of nodes in the source concept’s category tree that occur in the target concept’s category tree. The root node is not counted in the total number of nodes in the category tree. BLOOMS+ refines the inclusion measure by using a logarithmic function on the number of nodes and incorporates the depth of the common nodes. It is calculated as follows:

$$\text{Overlap}(T_i, T_j) = \frac{\log \sum_{n \in T_i \cap T_j} (1 + e^{d(n)^{-1}} - 1)}{\log 2 |T_i|}$$

where $$n \in T_i \cap T_j$$ are the common nodes between the source and target trees; and $$d(n)$$ is the depth of a common node $$n$$ in $$T_i$$. The exponentiation of the inverse depth of a common node gives less importance to the node if it is at a deeper level, and the log of the tree size avoids bias against large trees. It is stated that this equation ranges from 0.0 to 1.0 where 0.0 indicates no inclusion and 1.0 indicates maximum inclusion.

The contextual similarity between two classes is then computed to further support (or reject) an alignment. For class C from source ontology and class D from target ontology, BLOOMS+ calculates the contextual similarity by computing whether the superclasses of C and D are relevant. The similarity between each pair of trees based on the following equation:

$$Csim(T_i, T_j) = \frac{2R_c R_d}{R_c + R_d}$$

where $$R_c$$ (and $$R_d$$) are the fraction of superclasses of C and D supported by $$T_i$$ and $$T_j$$, respectively. A class s is supported by $$T_i$$ if its name matches a node in $$T_i$$ or if the Wikipedia article (or article category) corresponding to s matches a node in $$T_i$$. For computing the overall similarity of two classes, BLOOMS+ combines both the class
similarity generated by category tree overlap and the contextual similarity:

\[ O(T_i, T_j) = \frac{\alpha \text{Overlap}(T_i, T_j) + \beta \text{Csim}(T_i, T_j)}{2} \]  

(3)

where \( \alpha \) and \( \beta \) are weights for class similarity and contextual similarity.

Finally, BLOOMS+ examines all pairs of trees for class C and class D and selects the pair with the greatest \( O(T_i, T_j) \). If \( O(T_i, T_j) \) is greater than an established threshold, then a relationship is established between class C and D as follows:

– If \( O(T_i, T_j) = O(T_j, T_i) \), then BLOOMS+ sets C owl:equivalentClass D.
– If \( O(T_i, T_j) < O(T_j, T_i) \), then BLOOMS+ sets C rdfs:subClassOf D.
– Otherwise, BLOOMS+ sets D rdfs:subClassOf C.

4.2 WikiMatch

WikiMatch (Hertling and Paulheim 2012) is an OA system that like BLOOMS+ uses Wikipedia as an external knowledge source. Compared to BLOOMS+, however, WikiMatch’s method is simpler and takes less runtime. Wikipedia contains approximately 23 million articles which cover most every possible domain, though at varying levels of depth. These articles are written by volunteers around the world in 285 languages. Because Wikipedia articles exist in different natural languages and the same article in different languages are linked, WikiMatch can also be used to align ontologies in different natural languages.

For each ontology concept, Wikimatch extracts the following components: fragment, labels and comments in a string format. The Wikipedia search engine is used to search for these strings with stop-words removed. For each concept, WikiMatch generates a set of document ids for URI fragment \( S_f \), a set of documents ids for label \( S_l \),and a set of document ids for a comment \( S_c \). From those sets, the similarity of a source concept \( s \) and a target concept is computed by

\[ \text{max}_{S \in \{S_f, S_c, S_l\}} \frac{\#(S(s) \cap S(t))}{\#(S(s) \cup S(t))} \]  

(4)

where \( S(s) \) and \( S(t) \) is the set of document ids for each of the three respective components for
the source and target concept, respectively. The similarity of the source concept and target concept is derived from the similarity of their sets of documents for each of the three components. If the maximum similarity of those three components exceeds a certain threshold, it returns an equivalence mapping between the source and target concepts. WikiMatch uses two Wikipedia searches. The simple search approach uses the complete string and not the individual tokens in the component. WikiMatch also implements another approach called the individual token search approach where each individual token in the component is used to search Wikipedia. For the test cases reported in (Hertling and Paulheim 2012), the simple search approach had slightly better performances and took less runtime.

WikiMatch differs from BLOOMS+ in that it does not build a category tree but takes a much simpler approach and uses the set intersection and union on the document ids returned by the Wikipedia search engine for each concept. These sets are determined based on which component, i.e., fragment, label and comment is used to create the search strings passed to the Wikipedia search engine.

### 4.3 AgreementMaker

Unlike BLOOMS and BLOOMS+, AgreementMaker for LOD ontology alignment (Cruz et al. 2011) does not use Wikipedia as a background knowledge source for matching LOD reference ontology. Instead, it uses WordNet and other LOD reference ontologies as its background knowledge source. Several changes have been made to the traditional AgreementMaker alignment system to improve its ability to align LOD reference ontologies.

Tradition ontology alignment systems focus more on the equivalent relation between two concepts from two same or similar domain ontologies. However, since LOD reference ontologies have fewer equivalent relations, how to produce subclass and superclass relations is a challenge for aligning LOD ontologies.

For matching subclass and superclass relations, AgreementMaker applies an inference that if concept s in source ontology is equivalent to concept t in target ontology and if concept s’ from source ontology is a subclass of s, then s’ is also a subclass of concept t. AgreementMaker
applies several variations of this inference by using other LOD reference ontologies and WordNet. For example, if AgreementMaker finds that a concept $s$ from the source ontology has been defined as a subclass of an external or imported concept $r$ from another ontology, and a concept $t$ from the target ontology has been found equivalent to $r$, then $s$ is also a subclass of $t$.

For aligning LOD ontologies, AgreementMaker first uses one of its matchers, the Advanced Similarity Matcher (ASM). Basically, ASM calculates the string similarity between the strings which describe concept in source and target ontologies. These similarities calculated by ASM are compared with a predefined threshold parameter: if the similarity is greater or equal to the threshold, then AgreementMaker adds an equivalent mapping between the concepts to a set $S_{\text{equal}}$. Then it uses the inference techniques discussed before to generate sets $S_{\text{subclass}}$ and $S_{\text{superclass}}$. This basic approach has similarity with BLOOMS which applies the Jena reasoner to produce subclass and superclass mappings based on existing equivalence mappings.

### 4.4 Previous Experiments to Align LOD Ontologies

The Ontology Alignment Evaluation Initiative (OAEI) has different tracks, each with a set of reference alignments that are used to evaluate an OA system’s precision, recall, and f-measure. No track exists for aligning LOD ontologies; thus, no standard reference alignment exists for LOD ontologies. Researchers (Jain et al. 2010) had experts develop a benchmark, the LOD reference alignments between ontology schema pairs taken from eight LOD ontologies: AKT Reference (A), BBC program (B), DBpedia (D), FOAF (F), Geonames (G), Music (M), SIOC (S), and the Semantic Web Conference (W). The rationale for their selection is their substantial LOD coverage, domain diversity, and publicly available schemas. Experts produced subclass, superclass, and equivalence mappings between the concepts for the ontology pairs listed in Table 4.1. These LOD ontology pairs have been used by the two OA systems BLOOMS and AgreementMaker. Their performance results are shown in Table 4.1 and were taken from (Cruz et al. 2011). In (Cross et al. 2013) the LOD reference alignment pairs have been analyzed manually to determine what equivalence mappings exist for each pair.

<p>| Table 4.1 BLOOMS and AgreementMaker (AM) Results on LOD Reference Alignment | 11 |</p>
<table>
<thead>
<tr>
<th>LOD Pair</th>
<th># mappings</th>
<th>BLOOMS</th>
<th>AM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Prec</td>
<td>Recall</td>
</tr>
<tr>
<td>F, D</td>
<td>225</td>
<td>0.67</td>
<td>0.73</td>
</tr>
<tr>
<td>G, D</td>
<td>41</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M, D</td>
<td>645</td>
<td>0.39</td>
<td>0.62</td>
</tr>
<tr>
<td>W, D</td>
<td>519</td>
<td>0.70</td>
<td>0.40</td>
</tr>
<tr>
<td>M, B</td>
<td>528</td>
<td>0.63</td>
<td>0.78</td>
</tr>
<tr>
<td>S, F</td>
<td>22</td>
<td>0.55</td>
<td>0.64</td>
</tr>
<tr>
<td>W, A</td>
<td>366</td>
<td>0.42</td>
<td>0.59</td>
</tr>
</tbody>
</table>

BLOOMS has better recall except for F,D and G,D. For the F, D pair, there are two equivalence relations found and are based on exact string matching Person and Organization. For the G,D pair, BLOOMS does not find the only equivalence relation SpatialThing = Place which AM finds likely by one of its matchers that uses comment fields. SpatialThing includes the word ‘places.’

BLOOMS has a 0 result for both precision and recall because it is not able to find this equivalence mapping. For the G,D pair, BLOOMS is unable to infer any subclass and superclass mappings that AgreementMaker finds using this equivalence mapping. Of the five remaining pairs, AgreementMaker has much better precision only for the M,D pair. BLOOMS with Wikipedia finds more correct mappings but also finds more incorrect mappings; therefore, it has a much lower precision of 0.39.

5. WikiMatch for Aligning LOD Ontologies

WikiMatch was initially selected for modification because it was one of only two systems that already used Wikipedia as background knowledge in the OA process; however, it only used Wikipedia for determining equivalence mappings and could not determine subset/superset mappings. The developers of WikiMatch were eager to provide their source and to assist if any problems arose with using WikiMatch for the research to extend it to produce subclass/superclass mappings. The source for the other system BLOOMS+ using Wikipedia was available and was downloaded and a strong effort was put into trying to get it working; however, this effort was unsuccessful. Numerous attempts seeking assistance from the BLOOMS+ developer were made, but without any helpful responses, the decision was made to pursue implementing a different approach to determine subset/superset mappings using Wikipedia in WikiMatch.
Evaluation of the original WikiMatch was only done using the standard OAEI ontologies. No further research on or evaluation of WikiMatch's performance had been undertaken using actual LOD reference ontologies until this thesis research. Before any modifications were made to WikiMatch, an experiment was performed using WikiMatch to align the LOD reference ontologies used in (Jain et al 2010) and in (Cruz et al 2011) and described in section 4.4. This experiment demonstrated that WikiMatch’s current use of Wikipedia was not constructive in aligning LOD ontologies. The first section below describes the performance of the original WikiMatch on the conference track for the past two years of the OAEI competition and then on LOD reference ontologies. The second section describes the modifications made to improve WikiMatch’s performance on aligning LOD ontologies. Finally the last section describes the results of the modified WikiMatch on the LOD reference alignment ontologies.

5.1. Initial WikiMatch Experiments

In (Hertling and Paulheim 2012) the results of WikiMatch were reported on the OAEI conference track test case. Table 5.1 shows the results of WikiMatch in the conference track for both the OAEI 2012 and 2013 events. WikiMatch was proposed to use Wikipedia as background knowledge and could be used for aligning LOD ontologies; however, it had never been used to align LOD ontologies.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>f-measure</th>
<th>f-measure Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>OAEI 2012</td>
<td>0.7</td>
<td>0.45</td>
<td>0.55</td>
<td>8/18</td>
</tr>
<tr>
<td>OAEI 2013</td>
<td>0.7</td>
<td>0.45</td>
<td>0.55</td>
<td>11/22</td>
</tr>
</tbody>
</table>

The above table shows that WikiMatch’s performance did not change with respect to the performance measures between the two OAEI events. Its ranking over all, however, dropped as more and improved OA systems participated in the latest competition. In both years the best f-
measure was 0.71 and the worst F-measures were 0.36 (OAEI 2013) and 0.37 (OAEI 2012). WikiMatch’s f-measure falls in the middle of this range. These results are on equivalence mappings between classes within the ontologies being aligned.

Since WikiMatch had never been used to align LOD ontologies previously and to establish a baseline for comparisons after modifying WikiMatch as part of this thesis research, experiments were run using LOD reference alignments. Of the seven LOD ontology pairs in the LOD reference alignments five pairs are used: FOAF-DBpedia (F, D), Geonames-DBpedia (G, D), Semantic Web Conference-DBpedia (W, D), SIOC-FOAF (S, F), and Semantic Web Conference – AKT (W, A). Due to WikiMatch being unable to open the OWL version of the Music ontology, the two pairs with this ontology were not used in these experiments. Since the original WikiMatch is only able to produce equivalence mappings. Table 5.2 shows its performance if only equivalence mapping in the LOD reference alignments are used for evaluating its performance.

Table 5.2  Original WikiMatch on Five Pairs from the LOD Reference Alignments

<table>
<thead>
<tr>
<th>Test Pairs</th>
<th>Equivalence mappings in reference alignment</th>
<th>found</th>
<th>correct</th>
<th>precision</th>
<th>recall</th>
<th>f-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>F, D</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0.5</td>
<td>1.0</td>
<td>0.75</td>
</tr>
<tr>
<td>G, D</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, D</td>
<td>9</td>
<td>10</td>
<td>6</td>
<td>0.6</td>
<td>0.67</td>
<td>0.64</td>
</tr>
<tr>
<td>S, F</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1.0</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>W, A</td>
<td>7</td>
<td>13</td>
<td>3</td>
<td>0.23</td>
<td>0.43</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Although WikiMatch performs reasonably well on the conference track ontologies and can find some of the equivalence mappings in the LOD reference alignments, it does not have more sophisticated matchers to examine the structure of source and target ontology and techniques to produce subclass and superclass mappings between concepts in the source and target ontologies being aligned. Producing subclass and superclass mappings is essential to aligning LOD ontologies.

5.2 Adding Subclass/Superclass Mapping Capability to WikiMatch

To improve the performance of WikiMatch on LOD reference ontologies, the capability of
producing subclass/superclass mappings is needed. WikiMatch uses a similarity measure to
between the article sets for a source concept and target concept produce equivalence mappings.
The enhancement to WikiMatch is the use of an inclusion measure between the article setsto
produce subclass and superclass relations for the ontology alignment process.

The similarity measure used in WikiMatch, is the Jaccard index. Since this index is symmetric, it
cannot evaluate asymmetric relations such as subclass and superclass relations. To find subclass
and superclass mappings, an asymmetric measure is required, that is, a set inclusion measure.
The inclusion measure used between concept c1 in source ontology and concept c2 in the target
ontology is

\[ I(c1, c2) = \max_{S \in \{Sf, Sc, Sl\}} \left[ \frac{|S(t1) \cap S(t2)|}{|St(t1)|} \right] \]  \hspace{1cm} (5)

where Sf, Sc and Sl are defined as for the Jaccard index as given in equation 4, and t1 and t2 are
the string representations for the various components of concept c1 and c2 respectively. The
difference \( D \) between the two inclusion values is calculated as

\[ \text{Diff}_{inc} = I(c1, c2) - I(c2, c1) \]  \hspace{1cm} (6)

Based on WikiMatch 0.8 threshold for the Jaccard set similarity measure, the threshold for the
inclusion measure was set at 0.5. The rationale was that if half or more of a concept A’s article
set was included in concept B’s article set, that concept A is strongly related and in a subclass
role to B. However, a check has to be made that the inclusion of B’s article set in A’s article set
diffs by a significant amount, i.e., if the inclusion measures in both directions are close even if
with high inclusion values, then one could not say either concept is a subclass of the other. The
threshold for the difference in the inclusion measures is 0.1. Based on empirical observation
difference thresholds smaller than 0.1 include some concept pair which could be equivalent or
simply strongly related but with not a subclass or superclass mapping.

If the Jaccard similarity value \( S(c1, c2) \) is greater or equal to an equivalence mapping threshold
(0.8 as default of WikiMatch), then an equivalence mapping between c1 and c2 is added to the
ontology alignment result. Otherwise, the inclusion measure is calculated in both directions. If either of them is greater or equal to an inclusion threshold, and the absolute value $\text{Diff}_{inc}$ is greater than the difference threshold, a subclass or superclass mapping between $c_1$ and $c_2$ is added to the ontology alignment result. To determine whether it is a subclass or superclass relation, the sign of $\text{Diff}_{inc}$ is used. If $\text{Diff}_{inc}$ is greater than 0, $c_1$ is subclass of $c_2$, otherwise, $c_1$ is superclass of $c_2$. Adding this enhancement to WikiMatch required understanding the system's structure. Four different levels of data exist in WikiMatch:

1. Ontology model includes all entities and relations in ontology.
2. Resources are extracted from ontology model. Resources are the entities within the ontologies such as concepts and properties.
3. Subresources for each entity contains its name, label and comments.
4. Title sets which are retrieved from Wikipedia.

WikiMatch calls the Wikipedia API to search for a term of a concept (could be name, label or comment). It returns a set of titles of articles related to the search term. The following figure shows how WikiMatch handles this types of data.

Figure 2WikiMatchData Levels (Hertling and Paulheim 2012)

To modify WikiMatch to use the inclusion measure, a method is added to calculate the size of title sets for each subresource. These sizes are needed for the denominator of the inclusion measure.
Since the similarity calculation for the Jaccard index occurs at the subresource level, the inclusion measure is also calculated at the same point. The similarity measure and the inclusion measures for both inclusion directions are returned to the resource level. At the resource level, the threshold for the similarity measure is first checked to determine if an equivalence mapping exists. If the threshold is not met, then the inclusion threshold is checked. If the inclusion threshold is met, the difference in the two inclusion measures is determined. If the absolute difference meets the threshold, then the sign of the difference is checked in order to determine whether to produce a subset or superset mapping between the two concepts.

5.3 Experiments with Modified WikiMatch

The same five LOD reference alignment pairs as used for the original WikiMatch are used to evaluate the performance of the modified WikiMatch. The results of these experiments are presented in Table 5.3 and use all the mappings in the LOD reference alignments for evaluation purposes. The 0.5 and 0.3 indicates the results for the 0.5 and the 0.3 inclusion thresholds. The lower 0.3 threshold was additionally tried to see if the recall performance could be improved without lowering precision due to the very low recall values at the 0.5 threshold.

<table>
<thead>
<tr>
<th>Test Pairs</th>
<th>all mappings in reference alignment</th>
<th>Found 0.50.3</th>
<th>Correct 0.50.3</th>
<th>Precision 0.5 0.3</th>
<th>Recall 0.5 0.3</th>
<th>f-measure 0.5 0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>F, D</td>
<td>225</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>0.16</td>
<td>0.11</td>
</tr>
<tr>
<td>G, D</td>
<td>41</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, D</td>
<td>519</td>
<td>22</td>
<td>25</td>
<td>6</td>
<td>0.27</td>
<td>0.24</td>
</tr>
<tr>
<td>S, F</td>
<td>22</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>W, A</td>
<td>366</td>
<td>18</td>
<td>69</td>
<td>3</td>
<td>0.17</td>
<td>0.10</td>
</tr>
</tbody>
</table>

The result shows that now WikiMatch can find some of correct subclass and superclass relations. However, the F-measure result is not as good as expected. Lowering the threshold obvious finds more mappings but only for the S,F and W, A cases was it able to find more correct mappings. In the case of W, A it found many more mappings, almost 4 times as many mappings.

WikiMatch found 5 incorrect mappings for Geonames-Dbpedia, two of which are:

http://dbpedia.org/ontology/RugbyPlayer < 0.3597122302158273 http://www.geonames.org/ontology#Code
The concepts RugbyPlayer and RugbyLeague do not produce large article sets, and both have an article "dual-code rugby internationals" in their sets. This article contributes to these concepts being mapped to "code" since their inclusion measures are greater than the 0.3 threshold. This example demonstrates the drawback using an inclusion measure with Wikipedia: for terms having smaller article sets.

There are several factors related to its poor performance. WikiMatch only uses Wikipedia article titles for aligning LOD reference ontologies and does not consider the structures of ontologies or any more detailed information contained within Wikipedia. A number of LOD reference ontologies are multi-domain. When evaluating the similarity of two concepts, WikiMatch uses only the maximum similarity value among name, label and comment. This approach can lead to a problem if two concepts are in different domains but have same name. In addition, the sizes of the title sets produced by the Wikipedia API are small; some of the terms produced less than ten titles in their title sets. These small sizes of title sets make it challenging to calculate an accurate measure of inclusion. Because of these limitations, it was decided that a more sophisticated OA system should be selected for experimenting with Wikipedia as background knowledge to align LOD ontologies.

6. LogMap for Aligning LOD Ontologies

6.1 Background for Selection of LogMap

Early in the thesis research the source for BLOOMS+, the only other system using Wikipedia, was downloaded. After various debugging efforts and fixes to execution errors and numerous attempts seeking assistance from the BLOOMS+ developers were made, but without any helpful responses, the decision was made to pursue implementing the use of Wikipedia as background knowledge in another current OA system.

Initially, AgreementMaker was to be used since it has been a top performer in several of the recent OAEI competition although it did not participate in OAEI 2012. LogMap was a top
performing system in that competition. Another reason for initially selecting AgreementMaker was that two other master’s thesis students had used the system for their research and a reasonable collaboration avenue existed. Work had started on modifying AgreementMaker to implement the BLOOMS+ algorithm using Wikipedia. The other major issue was the evaluation of the addition of Wikipedia as background knowledge to AgreementMaker.

As discussed in section 4.4, the two systems, BLOOMS and AgreementMaker, had used a set of alignments referred to as the LOD reference alignments for evaluation purposes (Jain et al. 2010) (Cruz et al. 2011). The BLOOMS developers were contacted to obtain the LOD reference alignments. After several months of their not being able to locate the them, the AgreementMaker developers were contacted since they had also used the same LOD reference alignments. They immediately said they were unable to help. Finally another author on the paper (Cruz et al. 2011) was contacted, and he immediately provided the LOD reference alignments. A week later, Dr. Cruz emailed to state that these LOD reference alignments should not be used since they were not at liberty to provide them. After several emails back and forth questioning why this thesis research could not use the LOD reference alignments, Dr. Cruz abruptly rescinded Miami’s license agreement to user their OA system.

At this point, since LogMap was the top performer in several tracks of the OAEI 2013 competition, the developers of LogMap were contacted to inquire about using their OA system for experimenting with background knowledge to align LOD reference ontologies. They were extremely helpful and excited about this thesis research since although LogMap was a top performer, it made very little use of any background knowledge in its OA process.

6.2 LogMap Overview

LogMap (Jimenez-Ruiz and Grau 2011) implements highly optimal data structures for both lexically and structurally indexing the input ontologies. These structures are used to compute an initial set of anchor mappings and to assign a confidence value to each of them. The core of LogMap follows the general architecture for OA systems in that it uses an iterative process that starts from the initial anchors and then alternates the mapping repair and mapping discovery steps. In order to detect and repair unsatisfiable classes(Determining by propositional Horen
representation) on the fly during the matching process, LogMap implements a sound and highly scalable ontology reasoner as well as a `greedy' diagnosis algorithm. New mappings are discovered by iteratively `exploring' the input ontologies starting from the initial anchor mappings and using the extended hierarchies of the two input ontologies.

![Figure 2 LogMap nutshell (Jiménez-Ruiz and Grau 2011)](image)

The main steps of LogMap process are as following:

1. Lexical and structural indexation: LogMap indexes the labels of the classes in each ontology as well as their lexical variations, and allows for the possibility of enriching the indexes by using an external lexicon (e.g., WordNet or UMLS-lexicon). For structural indexing, LogMap uses an interval labeling schema (Nebot, 2009) to represent the extended class hierarchy of each input ontology. Each extended hierarchy can be computed using either simple structural heuristics, or an DL reasoner.

2. Computation of initial `anchor mappings': LogMap computes an initial set of equivalence anchor mappings by intersecting the lexical indexes of each input ontology. These mappings can be considered `exact' and later serve as starting point for the further discovery of additional mappings.

3. Mapping repair and discovery: In the repair step, LogMap uses a reasoning algorithm to detect classes that are unsatisfiable for both input ontologies and the current mappings computed so far. Then, each of these undesirable logical consequences is automatically repaired using a `greedy' diagnosis algorithm which recursively repair relevant classes by using the reasoning algorithm to detect unsatisfiable classes.

For discovering new mapping, LogMap maintains two contexts (One per ontology, consist of a set of classes determined by anchor and a subset of specific active classes for each iterations) for each anchor. Contexts for the same anchor are expanded in parallel using the class hierarchies of
the input ontologies by. New mappings are then computed by matching the classes in the relevant contexts by a flexible tool that computes a similarity score for any pair of input strings.

(4) Ontology overlapping estimation. In addition to the final set of mappings, LogMap computes a fragment of each input ontology, which intuitively represent the ‘overlapping’ between both ontologies. When manually looking for additional mappings that LogMap might have missed, curators can restrict themselves to these fragments since ‘correct’ mappings between classes not mentioned in these fragments are likely to be rare.

LogMap provides a sophisticated OA process, has participated in the last two OAEI competitions and was a top performer in both, has minimal use of background knowledge, and is open source; therefore, it serves as an ideal environment for experimenting with the use of background knowledge for aligning LOD ontologies.

Before starting modifications to LogMap and as done for WikiMatch, an experiment was performed as a baseline to see what its performance results would be on the LOD reference alignment. Table 6.1 shows these results and only for the equivalence mappings since LogMap does not produce the subclass/superclass mappings at this point. Producing these mappings is the objective of this thesis research.

Table 6.1  Original LogMap on Five Pairs from the LOD Reference Alignments

<table>
<thead>
<tr>
<th>Test Pairs</th>
<th>Equivalence mappings in reference alignment</th>
<th>found</th>
<th>correct</th>
<th>precision</th>
<th>recall</th>
<th>f-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>F, D</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>0.25</td>
<td>1.0</td>
<td>0.40</td>
</tr>
<tr>
<td>G, D</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, D</td>
<td>9</td>
<td>17</td>
<td>5</td>
<td>0.29</td>
<td>0.56</td>
<td>0.38</td>
</tr>
<tr>
<td>S, F</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>0.75</td>
<td>1.0</td>
<td>0.86</td>
</tr>
<tr>
<td>W, A</td>
<td>7</td>
<td>18</td>
<td>2</td>
<td>0.11</td>
<td>0.29</td>
<td>0.16</td>
</tr>
</tbody>
</table>

6.3 WikiNet

Both BLOOMS+ and WikiMatch use the Wikipedia API. WikiMatch uses a simple approach and only accesses titles for concepts found in source and target ontologies. BLOOMS/BLOOMS+ uses a more complex algorithm and builds multiple category trees for concepts found in the source and target ontologies. However, for aligning large-scale LOD ontologies like DBpedia, it
is very time consuming to use the Wikipedia API for each pair of concepts from source and target ontologies. This fact was confirmed by initial experiments using the Wikipedia API for building the category trees. It was decided that this very time-consuming approach would be too inefficient for participating in the OAEI competitions. BLOOMS/BLOOMS+ was a proof of concept system to investigate the use of Wikipedia. It has never participated in the OAEI competitions.

For this thesis research a more efficient way for exploiting Wikipedia knowledge was needed. WikiNet (Nastase and Strube 2010) is an organized and rich local source containing category and relation information about Wikipedia and is used as the background knowledge for this thesis research. WikiNet was built from Wikipedia’s existing network of categories and articles from the English language version of Wikipedia.

Wikipedia categories are noun phrases such as ALBUMS BY ARTIST that do not correspond to the type of lexical concepts typically found in texts and not of immediate use for processing a text automatically. For example, some category names are complex noun compounds such as MIXED MARTIAL ARTS TELEVISION PROGRAMS which consists of two noun phrases. This category name encodes an implicit relation ‘topic’ within its name, i.e., “television programs whose topic is mixed martial arts”. The first step of the WikiNet algorithm is to deconstruct category names by analyzing them and classifying into four different kinds base on the type of information they encode. As another example, the category ALBUMS BY ARTIST specifies a class attribute relation. The following figure illustrates this process for MIXED MARTIAL ARTS TELEVISION PROGRAMS category.
Figure 4 Relations and some instances induced for **Mixed Martial Arts Television Programs** (Nastase and Strube 2013)

After the deconstruction of category names, the network is further refined based on information in the articles’ infoboxes. The links between category names and information in the infoboxes are examined. The category name encodes the categorization/grouping criterion used by all of the subsumed pages. The infoboxes contain a summary of the most important information in the corresponding pages. The link between the information encoded in category names and the information summarized in infoboxes is used to propagate relations from infoboxes through the category network.

Finally this refined network is formalized. Nodes are considered to represent concepts and nodes considered as referring to the same concept are merged with each concept. Lexicalizations are added for concepts based on redirect, disambiguation and cross-language links from Wikipedia versions in different languages. Each concept is given a language independent ID. The result of this process is WikiNet with approximately 3.7 million concepts with close to 50 million relations for 454 relation types. Numerous evaluations have shown WikiNet to be a high quality resource and beneficial to various NLP tasks.

### 6.4 LogMap + WikiNet Algorithm

Since WikiNet is a multi-language large-scale knowledge source, the first step is extracting useful information from WikiNet in order to build category trees for source and target concepts. WikiNet provides several files that have been used to create a cached-based database using the Berkeley DB Java edition. An API is also provided as an interface to the database management system. Using this database management system necessitates the local installation of the Wikipet database/cache which requires more than 4GB. For this thesis research it was decided that building the category trees could be done simply by processing only a few of the files provided by WikiNet.

The index.wiki file is first processed to acquire concept IDs and then the data.wiki file is processed to the supercategories of a concept. The index.wiki contains all the names for concepts.
including those in other languages, and data.wiki contains all relations rather than only SUBCAT_OF relations. Thus, preprocessing is needed to extract the information needed for building the category trees. A python program is used to extract only the English name from the index.wiki file and only SUBCAT_OF relation is used from data.wiki. When the LogMap starts, the preprocessed index.wiki and data.wiki are loaded as HashMaps.

A category "forest" is a set of category trees for one ontology concept. To build a category tree for an ontology concept its class name must be retrieved based on its class id. The method `getProcessedName4ConceptIndex(int index)` in LogMap takes a class id as input and returns its class name as a lower case string. If there are "_" and space in the class name string, it tokenizes them into separate terms. Since all concept names in WikiNet index are in lowercase, the returned string can be used to search the hash table for index.wiki.

The category tree method takes the preprocessed class names to generate category "forest" for each ontology concept. The method `index.get(String name)` is called and returns the ID of each concept associated with the search string. Each of these IDs is used as a root node for a category tree. For each `ri`, the method `data.get(id)` is called as the input. This method searches the hash table for data.wiki and returns the subcategories for `ri`, using the entry `ri`, SUBCAT_OF ID1, ID2, … IDn where the IDs are supercategories of `ri`. Then for each ID1, ID2…IDn the method `data.get` is called to find its supercategories.

A depth limit of 4 is used for building the category tree as was done for BLOOMS based on its experiments. The experimental results showed that when the category tree goes beyond depth four, the super category nodes become very general. These general nodes lead to meaningless overlapping and, therefore, cause more incorrect mappings.

In WikiNet, an article and a category from Wikipedia may have the same concept id. For example, there is a ROME category and a ROME article. In data.wiki there could be a pattern ROME CATEGORY IDk1, IDk2, … IDkn which indicates the categories for the article ROME. The WikiNet concept id is used as the root node of the category tree and then the CATEGORY relation was used to find the article’s categories as its children. Then the SUBCAT_OF was used.
to build the category tree further down. This approach resulted in very poor performance results so it was decided to use only the SUBCAT_OF relation on the concept id that represents lexical variants of the search term.

For calculating the overlap confidence, the approach used by BLOOMS+ and described in section 4.1 is used. The contextual similarity measure as described in section 4.1 has been implemented and experiments were run combining it with the overlap measure. For these experiments, however, most of the contextual similarity measures produced were 0 so that only equivalence mappings were produced. The experiments reported here using the WikiNet algorithm, therefore, only uses the overlap measure given in equation 1 in section 4.1.

For the experiments, two thresholds of 0.50 and 0.75 are used. In (Jain et al. 2011) only a threshold of 0.50 was used, and in (Jain et al. 2010) a threshold of 0.6 or 0.8 was used depending on the LOD pair. The results reported in reported in Table 6.2 are those of (Jain et al. 2010). The AgreementMaker (AM) results are taken from (Cruz et al. 2011).

<table>
<thead>
<tr>
<th>LOD Pair</th>
<th>LogMap + WikiNet algorithm</th>
<th>AM</th>
<th>BLOOMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
<td>Recall</td>
<td>f-measure</td>
</tr>
<tr>
<td>F, D</td>
<td>0.50</td>
<td>0.75</td>
<td>0.50</td>
</tr>
<tr>
<td>G, D</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, D</td>
<td>0.08</td>
<td>0.18</td>
<td>0.04</td>
</tr>
<tr>
<td>S, F</td>
<td>0.63</td>
<td>0.75</td>
<td>0.23</td>
</tr>
<tr>
<td>W, A</td>
<td>0.07</td>
<td>0.17</td>
<td>0.02</td>
</tr>
</tbody>
</table>

From looking at Table 6.2 results, although the WikiNet algorithm added to LogMap was able to find some of the correct mappings for the LOD reference alignment pairs, its performance is quite poor as compared to that of BLOOMS and AgreementMaker except for the S,F pair. Also note that as the threshold increases that standard tradeoff between precision and recall occurs, i.e., precision increases but recall decreases. Simply using WikiNet as background knowledge does not greatly affect the ability of LogMap to align LOD ontologies. Further investigation of techniques used in BLOOMS and AgreementMaker were undertaken and the next several sections describes the addition of these techniques to LogMap.
6.5 LogMap + Inferencing on Equivalence Mappings

For aligning LOD ontologies, both BLOOMS and AgreementMaker use inferencing techniques to produce subclass and superclass mappings; BLOOMS uses post-processing with the Jena reasoner and Agreement maker uses its own inferencing method based its produced equivalence mappings. For example, if the OA system has already found an equivalence mapping between concept s from the source ontology and concept t from the target ontology, then if concept s’ from source ontology is a subclass (superclass) of s, then s’ is mapped as a subclass(superclass) of t. This same method is used to produce mappings between subclasses (superclasses) of concept t with respect to concept s. In (Cross et al 2013) a study was done to determine the extent to which inferencing methods contributed to the alignment results reported for LOD ontologies. After the disappointing results simply using WikiNet as background knowledge, the decision was made to add inferencing on equivalence mappings produced by LogMap.

LogMap indexes the class information from input ontologies using an interval labelling schema (Nebot and Berlanga 2009) which is an efficient data structure to store class relations as DAGs and trees. It uses two DAGs, the descendant DAG recording the descendants of a concept and the ancestor DAG for the ancestors. Each class C is represented as a node with a unique identifier in these two DAGs. Associated with each node are a descendant interval and an ancestor interval. The descendent (ancestor) interval specifies the range of concept ids that make up descendants (ancestors) for a concept. This interval labelling scheme significantly reduces the cost of executing typical queries like retrieving subclasses/super classes for a concept in a large-scale class hierarchy and makes implementing an inferencing method in LogMap very efficient.

After LogMap executes its process for producing equivalence mappings, the new inferencing method takes each pair (s, t) of an equivalence mappingand produces the inferred subclass (superclass) mappings. Two methods have been added to the LogMap method Classindex, getdirectedSubclasses() and getdirectedSuperclasses(). These two methods simply return the data members directedSubClasses and directedSuperClasses, respectively for the instance of Classindex. The subclasses (superclasses) for Classindex object C can be obtained by the method C.getdirectedSubclass() (C.getdirectedSuperclass()).
Figure 5 shows one level up and one level down inferencing for an(s, t) equivalence mapping produced by the LogMap system. With one level inferencing the first level up (down) concepts for source concept s only map as superclasses (subclasses) of the target concept t. Similarly, the first level up (down) concepts for target concept t have concept s mapped as their subclass (superclass). Here all mappings produced involve only the source and target concepts.

![Diagram of inferencing](image)

Figure 5 One-level inferencing

The two methods `getdirectedSubclasses()` and `getdirectedSuperclasses()` are called in the method `addinferencingrelation()` to generate subclass and superclass sets based on the parameter `sup`, a boolean variable that indicates which set of relationships to retrieve. The parameters `idea` and `ideb` are the concept IDs for the two concepts s and t that are considered equivalent. The parameter `infmaps` is a set of concept ids for either subclasses or superclasses based on the value for `sup`.

In Logmap2core class, the method `saveextractedmappings(string filename)` outputs mappings to file. It had been modified to output the subclass (superclass) mappings. The pseudocode is given below.

```plaintext
saveextractedmappings(string filename):
    OutPutFilesManager outPutFilesmanager
    for each idea which are in logmaps' candidate mappings:
        ...
```

27
for each ideb has mapped to idea:
  // add subclass and superclass mappings
  if idea < ideb: // when idea < ideb, idea is from source, ideb is from target
    if ideb and idea is in mapping list:
      // mapping list contains confirmed eq maps, which should be considered as equivalent maps to output
      // Now find superclasses and subclasses for idea concept and create additional mappings based // on inferencing using the equivalence relationship
      addinferencingrelation(outPutFilesmanager, idea, ideb, sup=true, infmaps)
      addinferencingrelation(outPutFilesmanager, idea, ideb, sup=false, infmaps)
      relation= EQ
      else relation=L2R

  // else, logmap consider it as a subclass relation idea<ideb. Because idea maps to ideb but ideb is not mapped to idea.
  write idea-relation-ideb to file
else // when idea > ideb
  if ideb and idea is in mapping list:
    // idea and ideb are swapped
    // Now find superclasses and subclasses for idea concept
    // and create additional mappings based
    // on inferencing using the equivalence relationship
    addinferencingrelation(outPutFilesmanager, idea, ideb, sup=true, infmaps)
    addinferencingrelation(outPutFilesmanager, idea, ideb, sup=false, infmaps)
  else write ideb-R2L-idea to file // reverse relation

Method addinferencingrelation is considered as a helper method for this saveextractedmapping method. It is called within the double loop and only for confirmed LogMap equivalence mappings.

addinferencingrelation(OutPutFilesManager outPutFilesManager, int idea, int ideb, Boolean sup, Set<int[]> infmaps)

Set<Integer> set; // for containing ideb's sub/superclass'id
int relation; // logmap uses integer for representing relation
if sup==true://if sup is true it means it is going to map idea to ideb's subclass
  set = index.getClassIndex(ideb).getDirectSubclasses():// get ideb's subclass
  relation = Utilities.R2L:// idea<ideb'subclasses
else:// if sup is false it means it is going to map idea to ideb's superclass
  set = index.getClassIndex(ideb).getDirectSuperclasses();
  relation = Utilities.L2R:// idea<ideb'superclasses
if set is empty, do nothing;
else:
  for each cid in set:
    if (cid>=idea):// idea is from source, cid from target
int array map = {idea,cid,relation};
if map has already in infmaps, do nothing;//already output this map
else:
    output map to the mapping file;
    add map to infmap;
else://cid<idea
    cid from source, idea from target, for keeping map in same
direction, need to swap idea and cid, and reverse relation
    reverse relation;
    int array map = {cid, idea, relation};
    if map has already in infmaps, do nothing;
    else:
        output map to the mapping file;
        add map to infmap;

After implementing the one level inferencing within LogMap, an experiment using the LOD reference alignments was performed. The results are shown in Table 6.3. The results reported for BLOOMS (Jain et al 2010) and for AgreementMaker (Cruz et al. 2011) are included for comparison purposes.

Table 6.3 LogMap + Level 1 Inferencing Results for LOD Reference Alignments

<table>
<thead>
<tr>
<th>LOD Pair</th>
<th>LogMap + WikiNet algorithm</th>
<th></th>
<th></th>
<th>AM</th>
<th></th>
<th></th>
<th>BLOOMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
<td>Recall</td>
<td>f-measure</td>
<td>Prec</td>
<td>Recall</td>
<td>f-measure</td>
<td>Prec</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td>0.75</td>
<td>0.50</td>
<td>0.75</td>
<td>0.50</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>F, D</td>
<td>0.20</td>
<td>0.30</td>
<td>0.04</td>
<td>0.01</td>
<td>0.12</td>
<td>0.16</td>
<td></td>
</tr>
<tr>
<td>G, D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.26</td>
<td>0.68</td>
<td></td>
</tr>
<tr>
<td>W, D</td>
<td>0.08</td>
<td>0.18</td>
<td>0.04</td>
<td>0.01</td>
<td>0.06</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>S, F</td>
<td>0.63</td>
<td>0.75</td>
<td>0.23</td>
<td>0.14</td>
<td>0.43</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>W, A</td>
<td>0.07</td>
<td>0.17</td>
<td>0.02</td>
<td>0.01</td>
<td>0.05</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.48</td>
<td>0.43</td>
<td>0.45</td>
<td>0.42</td>
<td>0.59</td>
<td>0.50</td>
<td></td>
</tr>
</tbody>
</table>

As above table shows, this one level inferencing method performed very well for the S, F pair, as compared to AgreementMaker and BLOOMS. It actually outperformed AM on precision and tied on recall, thus giving it a higher recall. BLOOMS, however, outperformed on recall. The one level inferencing was successful for the S,F pair since both are small-scale ontologies and not very deep class hierarchies. For example, it found all the subclass relation between "Document" from source and specific documents such as "post" and "image" which are subclasses of "Document" in target.

For all other pairs, although the precision was fair, the recalls produced were quite low when compared to the other two BLOOMS and AgreementMaker. After analyzing the missed correct mappings and correct mappings produced by one level inferencing method, it was found that
many of the mappings in the LOD reference alignments require two level inferencing. The one level inferencing method can only produce a mappings between a class to the direct subclasses(superclasses) of its equivalent class. For example, in the F, D pair, the class AGENT in FOAF is a superclass of PERSON which is mapped as an equivalent to PERSON in DBpedia; therefore, AGENT in FOAF is a superclass of PERSON in DBpedia. PERSON in DBpedia is a superclass of specific role for a person such as SCIENTIST, POPE, etc.. The one level inferencing method cannot map class AGENT in FOAF as a superclass of the 50 different subclasses of PERSON in DBpedia.

![Diagram](image)

Figure 6 AGENT to group of person example

To improve the performance of the inferencing method, a two level inferencing technique has been implemented. Figure 7 shows first adding the additional levels of grandparents and grandchildren to the inferencing scheme.
First with two level inferencing in addition to the first level up (down) concepts, the second level up (down) concepts are also mapped as superclasses (subclasses) of the source and target concepts. Then for the second level inferencing method, the first level up (down) concepts not only map as superclasses (subclasses) of the source and target concepts but also map as superclasses (subclasses) to the children (parent) and grandchildren (grandparents) of the source and target concepts. The second level up (down) concepts not only map as superclasses (subclasses) to the source and target concepts but also map as superclasses (subclasses) to the children (parents) of the source and target concepts.

The second level up (down) concepts were initially mapped as superclasses (subclasses) to the grandchildren (grandparents) of the source and target concepts, but after experiments produced disappointing precision results and did not improve recall, those mappings are no longer produced by the second level inferencing method. This final mappings produced are illustrated in Figure 8.
The getdirectedSubclasses() and getdirectedSuperclasses() method are modified by adding the superclasses and subclasses of the equivalent classes for a specified class into the result set: The following describes the changes made to these methods.

getdirectedSubclasses(int id)/getdirectedSuperclasses(int id):

set resultset;
eqset = getequivalentclasses(id);
eqset.add(id)
for cid in eqset:
    resultset.addAll(getequivalentclasses(cid));
return resultset;

The subclass/superclass sets are enriched with the subclasses(superclasses) of equivalent classes within the same ontology by calling LogMap classindex method getEquivalentclasses(). The addinferencingrelation() method, was modified as follows:
addinferencingrelation(OutPutFilesManager outPutFilesmanager, int idea, intideb, Boolean sup, 
Set<int[]>infmaps)

    Set<Integer> set; //for containing ideb's sub/superclass'id
    Set<Integer> aset; // for containing idea's sub/superclass id
    int relation; //logmap uses integer for representing relation
    if sup==true://if sup is true it means it is going to map idea to ideb's subclass
        set = index.getClassIndex(ideb).getdirectSubclasses();//get ideb's subclass
        for each cid in set:
            // add all level two subclasses to set
            set.addAll(index.getClassIndex(cid).getdirectSubclasses());
            //add idea's superclasses and equivalent classes to aset which will map to set.
            aset.addAll(index.getClassIndex(idea).getdirectSuperclasses());
            aset.addAll(index.getClassIndex(idea).getEquivalenterclasses())
            relation = Utilities.R2L;//idea>ideb'subclasses
    else:                // if sup is false it means it is going to map idea to ideb's superclass
        set = index.getClassIndex(ideb).getdirectSuperclasses();
        for each cid in set:
            set.addAll(index.getClassIndex(cid).getdirectSuperclasses());//get level2
            aset.addAll(index.getClassIndex(idea).getdirectSubclasses());
            aset.addAll(index.getClassIndex(idea).getEquivalenterclasses());
            relation = Utilities.L2R;// idea<ideb'superclasses
        if set is empty, do nothing;
        else:
            for each id in aset:
                for each cid in set:
                    if (cid>=idea)://idea is from source, cid from target
                        int array map = {idea,cid,relation};
                        if map has already in infmaps,do nothing; //aready output this map
                        else:
                            output map to the mapping file;
                            add map to infmap;
                    else://cid<idea   cid from source,idea from target, for keeping map in same
                    if map has already in infmaps,do nothing;
                    else:
                        output map to the mapping file;
                        reverse relation;
                        add map to infmap;
            Table 6.4 shows the results of this two-level inferencing method and for ease of comparison
            repeats the results for AgreementMaker and BLOOMS.
Table 6.4 LogMap + Level 2 Inferencing Results for LOD Reference Alignments

<table>
<thead>
<tr>
<th>LOD Pair</th>
<th>LogMap + 2-level inferencing</th>
<th>AM</th>
<th>BLOOMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
<td>Recall</td>
<td>f-measure</td>
</tr>
<tr>
<td>F, D</td>
<td>0.43</td>
<td>0.71</td>
<td>0.57</td>
</tr>
<tr>
<td>G, D</td>
<td>0.26</td>
<td>0.68</td>
<td>0.47</td>
</tr>
<tr>
<td>W, D</td>
<td>0.38</td>
<td>0.20</td>
<td>0.33</td>
</tr>
<tr>
<td>S, F</td>
<td>0.44</td>
<td>0.39</td>
<td>0.41</td>
</tr>
<tr>
<td>W, A</td>
<td>0.71</td>
<td>0.45</td>
<td>0.58</td>
</tr>
</tbody>
</table>

In general, though the precision has been lowered, this two level inferencing method improved the recall greatly and, therefore, the f-measure. For the S, F pair the two level inferencing improved both precision and recall which are higher than AM. Its recall, however, is still lower than that of BLOOMS. Due to the improved two level inferencing, two more correct mappings were produced. For the F,D pair, the improvement is most significant as compared to the one level result in Table 6.1. The recall improved to 0.71 from 0.12. It finds most correct mappings excepts the "Spatialthing" class from FOAF to subclasses of "Place" from DBpedia as mentioned before. This result is because LogMap cannot produce the equivalence mapping between SPATIALTHING and PLACE. This problem is more significant for the W, A and the W, D pairs since LogMap only finds 2 of 7 equivalent mappings for W, A and 5 of 9 for W, D. Because these equivalence mapping are not found, the inferencing method cannot produce correct subclass(superclasses) mappings. Moreover, the incorrect equivalence mappings LogMap causes the two level inferencing to produce more incorrect subclass and superclass mappings. The result is a lower precision for all pairs as compared to the one level inferencing method.

The two level inferencing does provide LogMap with some capability to perform alignment on LOD ontologies that it did not originally have. This process does not make use of any background knowledge for these experiments since the effect of inferencing on LOD ontology
alignment needed to be investigated before combining it with the use of background knowledge. Both AgreementMaker and BLOOMS results in Tables 6.1 and 6.2 are produced with the use of background knowledge, respectively WordNet and Wikipedia. The following section describes the results of integrating the use of background knowledge with two level inferencing to align LOD ontologies.

6.6 LogMap + Inferencing on Equivalence Mappings + WikiNet Algorithm

The two approaches added to LogMap, two level inferencing on equivalence mappings and using WikiNet as background knowledge, were first experimented with individually to investigate their performance on aligning LOD ontologies. The integration of these two approaches is described and the results of aligning the 5 pairs of LOD ontologies are presented in Table 6.4. The thresholds used for the WikiNet measure to produce a subclass/superclass mappings are 0.5 and 0.75 as was used previously.

Several approaches were taken to integrating the Wikinet algorithm into LogMap. LogMap produces candidate equivalence mappings, each with confidence levels. The first approach is to examine these candidate mappings to determine if those with low confidence might be superclass or subclass mappings by using the WikiNet algorithm. However, through experimentation is as determined that the number of candidate mappings are very limited so this approach was not used but instead all pairs of concepts between the source and target ontologies are input to the WikiNet algorithm. The mappings produced by the WikiNet algorithm are then integrated with those produced by the two level inferencing algorithm.

For integrating the result produced by WikiNet algorithm and inferencing method, two problems are addressed.

1. Redundancy: Since these two approaches are working separately, the same mapping might be produced by both. For avoiding the redundancy, the two results are merged and only one of the mappings is included in the final result.
2. Confliction: If there are two classes having differing mappings produced by WikiNet algorithm and inferencing method, a confliction occurs. Based on previous experimental results showing that inferencing has better performance than the WikiNet algorithm, the mapping produced by inferencing method is kept. This confliction algorithm is included but from experimental results, no conflictions occurred.

Table 6.5 LogMap + Level 2 Inferencing + WikiNet Algorithm

<table>
<thead>
<tr>
<th>LOD Pair</th>
<th>LogMap + 2-level inferencing + WikiNet algorithm</th>
<th>AM</th>
<th>BLOOMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
<td>Recall</td>
<td>f-measure</td>
</tr>
<tr>
<td>F, D</td>
<td>0.50</td>
<td>0.75</td>
<td>0.50</td>
</tr>
<tr>
<td>G, D</td>
<td>0.40</td>
<td>0.43</td>
<td>0.71</td>
</tr>
<tr>
<td>W, D</td>
<td>0.23</td>
<td>0.35</td>
<td>0.24</td>
</tr>
<tr>
<td>S, F</td>
<td>0.67</td>
<td>0.71</td>
<td>0.55</td>
</tr>
<tr>
<td>W, A</td>
<td>0.34</td>
<td>0.43</td>
<td>0.40</td>
</tr>
</tbody>
</table>

Table 6.5a Two level Inferencing Results for Comparison

<table>
<thead>
<tr>
<th>LOD Pair</th>
<th>LogMap + 2-level inferencing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
</tr>
<tr>
<td>F, D</td>
<td>0.43</td>
</tr>
<tr>
<td>G, D</td>
<td>0.00</td>
</tr>
<tr>
<td>W, D</td>
<td>0.38</td>
</tr>
<tr>
<td>S, F</td>
<td>0.71</td>
</tr>
<tr>
<td>W, A</td>
<td>0.44</td>
</tr>
</tbody>
</table>

As Table 6.5 and Table 6.5a show, the WikiNet algorithm increases recall over the two level inferencing method for all pairs when the threshold is set at 0.5. For the 0.75 threshold the threshold results are the same as using only two level inferencing. For precision at the 0.75 threshold, the results are almost the same. For the 0.50 threshold, the pairs with the SWC (W) ontology have more significant decreases in precision.

When comparing the two level inferencing and WikiNet algorithm to the BLOOMS and AgreementMaker results, except for the S, F pair which has better f-measure results for both thresholds than AgreementMaker and slightly better for 0.5 threshold and slightly worse for the 0.75 threshold than BLOOMS, the modified LogMap systems does not perform as well. One of the problem pairs in particular is the W,D pair with f-measures less than half of BLOOMS f-
measure and substantially less than AgreementMaker’s f-measure. The modified LogMap produces substantially lower precision values on the pairs. Some possible reasons for this are

(1): The WikiNet algorithm uses the preprocessed class name provided by LogMap as a search term. WikiNet’s index is searched to find a concept matching that term. However, the preprocessing LogMap provides can only deal with single word. For some term like "OnlineAccount", this process can only produce a lowercase string "onlineaccount" which cannot find any matched concept in WikiNet index.

(2) Unlike BLOOMS which executes an online search for each pair of concepts using Wikipedia API, the WikiNet algorithm uses the WikiNet index file. The Wikipedia API returns a set of article related to those search terms. The WikiNet algorithm produces the concepts which are lexical variants of the search terms. This search process is not identical to the BLOOMS approach since there is not the capability to find related articles. The difference in the search method using the source and target class names may cause the WikiNet algorithm to miss concepts related to search terms. The result is the WikiNet algorithm produces category "forests" which are not as comprehensive as those of BLOOMS.

(3) For some class names which have a high generality such as "person", since each node of their category trees is general, they are likely to be mapped to other general class names. The occurrence of very general terms in a category tree produces higher overlapping of such category trees and this higher overlap leads to incorrect mappings. For example for the W, D pair, PERSON is mapped to COUNTRY, TAX, and TOWN by the WikiNet algorithm since they are so general that most of their category trees are overlapped. As Figure 9 shows, class COUNTRY in DBpedia has a super category HUMAN GEOGRAPHY which is at the first depth in its category tree. As a subtree in COUNTRY, HUMAN GEOGRAPHY makes up more than half proportion. HUMAN GEOGRAPHY is also in the category tree for PERSON from FOAF. These two category trees have a large overlap which leads these a subclass relation being established between PERSON and COUNTRY.
7. Conclusions and Future Research Directions

In this thesis, current research on aligning LOD ontologies is overviewed. Two current OA systems are modified to perform alignment of LOD ontologies using Wikipedia as a background knowledge source. For each modified OA system, experiments have been performed using a set of LOD reference alignments to evaluate their alignment results using standard OA performance measures.

For the modified WikiMatch system, an inclusion measure was added to produce subclass and superclass mappings between source and target concepts. The experiment results shows that using an inclusion measure between the article sets associated with the source and target concepts does not perform well for LOD ontologies alignment although the results for the S,F pair are fair since this pair has the fewest mappings in its reference alignment.

For the modified LogMap system, a local data source WikiNet is used for replacing the online searching in Wikipedia. An inferencing method based equivalent mappings produced by LogMap has also been added. For the S,F pair in the five LOD reference alignment cases, it performs better than AgreementMaker and BLOOMS. However the two added algorithms to LogMap still have limitations. Using a local resource for acquiring Wikipedia category information, unlike using Wikipedia API causes some of the search strings to go unmatched, and, therefore the modified LogMap misses correct mappings between the source and target concepts. The performance of inferencing method is partially determined by the performance of LogMap on producing correct equivalence mappings. If LogMap misses a correct mapping or produces an incorrect mapping, the inferencing method can also miss a correct subclass or superclass.
mapping or produce an incorrect subclass or superclass mapping.

For the limitations for using WikiNet, several possible future research work can be undertaken:

1. Since Wikipedia API has high performance on searching on strings referring to concept names, a combined process might be used. First the Wikipedia API could be used to generate the roots of the category trees instead of using WikiNet’s index file. Then WikiNet data file could be used for building the category hierarchy trees. In this way, it not time consuming as building the complete category tree through Wikipeida API calls but could perform better by finding more category trees, i.e., build a larger forest, for a concept.

2. A WikiNet toolkit has been developed for exploiting the resource on WikiNet. It provides a database management system which much functionality so that more WikiNet content could be used to improve the WikiNet algorithm for aligning LOD ontologies

3. Since LOD ontologies contain number of concepts which are referenced from other LOD ontologies which are not considered in the reference alignments, traditional OA system may produce equivalent mappings between them which lead to incorrect subclass/superclass mappings. A URI filter may be needed after OA system produce equivalence mappings.
Reference


Nebot, V. and Berlanga, R., Efficient retrieval of ontology fragments using an interval labeling scheme, Information Sciences, 179 4151–4173.2009
