Extending an Ontology Alignment System with a Lexical Database

Baththama Bello Alhassan
Department of Mathematics
Ahmadu Bello University
Zaria, Nigeria

Sahalu B. Junaidu
Institute of Computing & ICT
Ahmadu Bello University
Zaria, Nigeria

A.A. Obiniyi
Department of Mathematics
Ahmadu Bello University
Zaria, Nigeria

***Abstract*- Ontology alignment is a challenging problem which limits or even prevents interoperability among information systems. It aims to find semantic correspondences between a pair of input ontologies with a view to making them semantically interoperable. One product of ontology alignment research is Falcon Aligning Ontologies (Falcon-AO), a system that attempts to align a pair of input ontologies automatically. This paper developed and implemented an enhanced version of Falcon-AO with increased precision. The enhanced system makes use of a large lexical database, WordNet, in the alignment process. Performance of the extended Falcon-AO system was tested and validated using standard benchmark ontologies from the Ontology Alignment Evaluation Initiative. Comparative performance analysis of the original and extended.**

Falcon-AO systems based on alignment precision and execution time shows the extended system to be more precise, although at a small execution time penalty.

***Index Terms*— Ontology Alignment; WordNet**

I. Introduction

With the increase use of the World Wide Web (WWW) for communication and information exchange, the need for semantic interoperability is growing due to the heterogeneity of information. Ontologies are key components of semantic interoperability.

Ontologies express the structure of domain knowledge and enable knowledge sharing. Various domains have many ontologies that were designed by different domain experts from various points of views. The available ontologies are themselves sources of heterogeneity (2). Given two ontologies, the same entity can be given different names or simply be defined in different ways, or both ontologies may express the same knowledge but in different languages. This can be solved by mapping or aligning ontologies i.e. finding the correspondence between their components (Classes, Properties and Objects).

Ontology alignment is the major approach to solve the heterogeneity problem of information between heterogeneous ontologies in semantic web by extracting correspondences between pairs of ontologies. Alignment consists of a set of correspondences between their entities. A formal definition of alignment from (8) is as follows:

Let O, O' be two ontologies. Matching O with O' finds a set of mappings Each is a 5-tuple: $<id, e, e', r, sim>$ where is a unique identifier, is an entity in , is an entity in , is an equivalence or disjoint relationship that holds between and , and is a confidence value (similarity) between and e' in the range .

Ontology alignment is used for several applications such as ontology engineering, information integration, peer to peer information sharing, web service composition and query answering on the web. Ontology alignment can help solve the problem of semantic heterogeneity and semantic interoperability between different web applications and services.

Recently, many ontology matching systems have been developed such as Falcon-AO, TaxoMap, ASMOV, Anchor-Flood, and AROMA where each of these extracts one or more aspects of similarities between different ontologies using different strategies.

There are different strategies for finding similarities between entities in existing ontology matching systems. These strategies could be string similarity, structural similarity and strategies based on instances. As stated in (3) most of the ontology alignment approaches use syntactic matching techniques which map elements by analyzing entities in isolation, ignoring their relationships with other entities. Some have considered the structural matching technique which finds similarities among entities in structural graphs of two ontologies while some systems have considered the use of external background knowledge as a way to obtain semantic mappings between syntactically dissimilar ontologies.

Two limitations of current ontology alignment approaches that only rely on string and structure similarity (i.e., syntactic approaches), are that they do not provide semantic mappings and that they fail to discover some correct mappings when the

mapped ontologies have weak or dissimilar structures. Even though, in many cases, syntactic approaches can denote meaningful mappings, it is not always certain. By ignoring semantics, syntactic techniques fail to identify several important mappings (16).

This paper addresses the problems mentioned in the last paragraphs in an ontology alignment system, by incorporating WordNet into the system to align synonymous terms in the ontologies.

The paper is structured as follows. Related work is discussed in Section 3. Section 4 presents the proposed system and Section 5 describes the implementation of the extended Falcon-AO. Section 6 outlines and presents the experimental set up while summary and future work are presented in Section 7.

II. WordNet Strategy

Synonyms strategy can help in solving the problem of using different terms in the ontologies for the same concepts. For example, one ontology might use "Feast" while the other ontology is using "Banquet" for the same meaning. More often, synonyms strategy is based on external resources like thesaurus (e.g., WordNet, Wiki-dictionary), domain ontology, and corpus.

WordNet consists of a set of synonyms "synsets" which denotes a concept or a sense of a group of terms. Synsets provide different semantic relationships such as synonymy (similar) and antonymy (opposite).

III. Related Works

A. Automatic Semantic Matching of Ontologies with Verification (ASMOV) **(**15) as an iterative process incorporates two components similarity calculation and semantic validation process which removes any incorrect or invalid matches. It takes as input two ontologies in OWL-DL and an optional input alignment. It also uses several sources of general and domain specific background knowledge, such as WordNet and UMLS, to provide more evidence for similarity computation (4).This pre-alignment undergoes a process of semantic verification to eliminate correspondences that are less likely to be suitable based on the information present in the ontologies. Semantic verification in ASMOV investigates five types of patterns, e.g. Multiple-entity correspondences, crisscross correspondences, crisscross correspondences, domain and range incompleteness and disjointness-subsumption contradiction. Then produces an output of n: m alignments.

ASMOV is not a generic system; it has problems very large ontologies (11) Semantic verification in ASMOV eliminates too many alignments and requires too much time, thus leading to inefficiency (9) while the proposed system (Extended Falcon-AO) uses partitioning technique for large ontologies which reduces the amount of memory usage and time of execution.

B. **Anchor-Flood** (10) is an ontology alignment system that aims at achieving high performance. Input ontologies are ontologies represented in RDFs and OWL. This algorithm starts off with an anchor which is a pair of concepts with an exact string match of concepts, properties or instances pair from each ontology, gradually exploring concepts by collecting neighboring concepts i.e., superconcept, subconcept and siblings, thereby creating small segments from the ontologies to be matched. Then a segment-to-segment matching is done using string and structural similarity measure to produce an output of 1:1 alignments.

From (9) Anchor-Flood does not consider the entire ontologies because it does segment-to-segment comparisons only, thus reducing the system scalability. Semantic similarity between entities is not explored. Only exact string matches of concepts, properties and instances are considered as an anchors which leads to inefficient alignments. In our extended Falcon-AO system, the whole ontology needs to be processed to find anchors.

C. **Association Rule Ontology Matching Approach** (AROMA) (1) s matcher invented to find relations of equivalence and subsumption between entities, i.e. classes and properties of two OWL ontologies. It relies on the rule:

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

The overall process of AROMA is divided into three main successive stages: The pre processing stage starts by representing each entity, i.e. classes and properties, by a set of terms. The second stage consists of the discovery of association rules between entities, and then the post processing stage aims at cleaning and enhancing the resulting alignment by performing deduction of equivalence relations, suppression of cycles in the alignment graph, suppression of redundant correspondences, selection of the best correspondence for each entity the enhancement of the alignment .

AROMA is a time efficient matcher and like the extended Falcon-AO takes less memory space in finding alignments. But as stated in (11) the precision of AROMA is degraded in some ontology tracks due to the subsumption correspondences it returns**.**

D. **TaxoMap** (6) is an ontology matching algorithm designed to align two OWL ontologies consisting of two partitioning algorithms namely: Anchor Partition Partition (APP) and Partition Anchor Partition (PAP) which have been designed to take the alignment objective into account in the partitioning process. The most structured ontology is referred as target ontology and the less structured is referred as source ontology. PAP is suitable for structured against unstructured ontology matching and APP is suitable for structured against structured ontology matching.

Anchor Partition Partition (APP):

1. Find the set of anchors across ontologies.
2. Partition both the target ontology and source ontology by modifying PBM matcher in order to take into account shared anchors.
3. Align blocks that share maximal number of anchors.

Partition Anchor Partition (PAP):

1. Use PBM matcher to partition the target ontology into set of blocks and making the

source ontology to follow the pattern of the target ontology.
2. Identify the set of anchors between two ontologies. This set will be the center of the future block which will be generated from the source ontology.
3. Use PBM matcher to partition the source ontology around the identified centers.
4. Align each block with the corresponding block.

The drawbacks of this method, according to (9) are as follows. The effectiveness of this method depends on the availability of identical labels across ontologies. Only labels and hierarchy structure is used for matching and hence comparatively less recall. Apart from string and structural similarity extended Falcon-AO uses virtual document matcher which creates virtual documents for every entity and Similarity between two entities is determined by the similarity between their virtual documents.

E. Falcon-Aligning Ontologies (7): Measures string and structural similarities using Partition Based Block Matching, Virtual Document, I-Sub and Graph Matching for ontologies. It is quite a flexible ontology alignment system and time economic. It is one of the most popular choices for matching web ontologies; it is free and open source. Unfortunately, Falcon-AO has its draw backs in cases when there is little or no lexical overlap in the entities of the ontologies and when the structures of the two ontologies are different Falcon-AO cannot detect matching. It also uses only string and structural matching techniques.

This paper aims to extend Falcon-AO to find synonymous terms in two different ontologies and combine alignments to the final result. The extension is achieved through the use of WordNet, a lexical knowledge source. Extended Falcon-AO preserves input ontologies and matching techniques as Falcon-AO. It considers all alignments by WordNet correct, with similarity value 1.0. The extended Falcon-AO runs a little slower than Falcon-AO. The comparative analysis revealed that the extended Falcon-AO is favored in terms of precision, at the expense of bearable additional execution overhead.

The extended system is realized by adding a Synonym Search layer to the architecture of the original Falcon-AO system as shown in

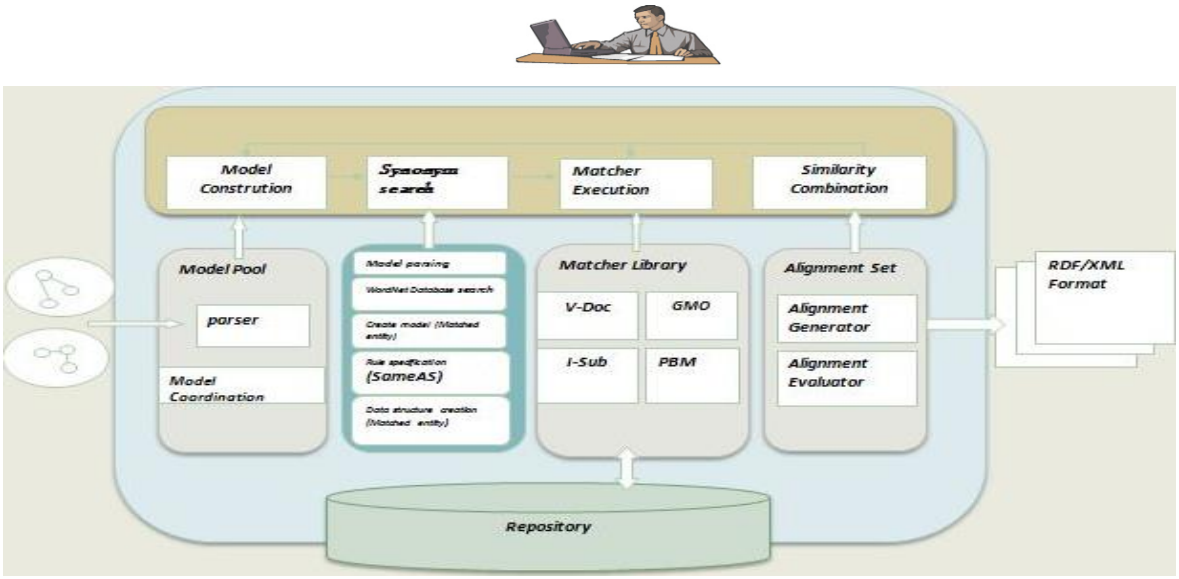

Figure 1: Extended Falcon-AO System

The six major components in the system are:

A. ***The Model Pool*** which parses input ontologies into models (in memory) by Jena and adjusts models by using a set of coordination rules.

B. ***The Matcher Library*** handles a collection of matchers which are the Virtual Document (V-Doc), I-Sub, Graph Matcher for Ontologies (GMO) and Partition Based Block Matching (PBM).

- **I-Sub** (5): - It is a linguistic Matcher simply based on the string comparison techniques. Its novelty is not only that the commonalities between the descriptions of domain entities are calculated but also their differences are examined.
- **V-Doc** (14): - Is also a linguistic matcher that discovers alignment by exploiting RDF graph of ontology to create a *virtual document* of each entity in the ontology which contains the local description and also the neighboring information to reflect the intended meaning of the entity
- **GMO** (12): -Is a structural matcher that utilizes RDF bipartite graphs to represent ontologies and computes structural similarities between domain entities and between statements in the ontologies.
- **PBM** (13) - Normally, large ontologies pose significant challenges to the present day ontology matching systems. PBM utilizes the divide and conquer approach to find block matchings.

C. ***Alignment Set:*** Generates alignments by using a widely-accepted RDF/XML format and evaluates

generated alignments against reference alignments based on the conventional precision/recall metrics.

D. ***Central Controller***: - Executes matchers. How does the system combine the similarities from the matchers for ontologies? The central controller combines similarities based on measures of linguistic comparability and structural comparability.

The linguistic comparability:-is computed by examining the proportion of the candidate alignments against the minimum number of domain entities in the two ontologies.

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

E. ***Repository:*** stores reusable data during the matching process.

IV. IMPLEMENTATION

In the proposed approach for matching ontologies based semantic similarities using WordNet, five more sub-components were added to find semantically corresponding terms between two given sources. First step is the pre-processing step; the second step is the WordNet search, the third is the model creation, the forth is the rule specification and the final step is the data structure and merging.

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

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

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

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

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

F. VI. Experiment and Result

To evaluate the performance of extended Falcon-AO, benchmark conference track test data taken from OAEI were used. The OAEI is widely used for ontology matching evaluation. It does not only assess the strength and weakness of matching methods but also helps improve the work by providing test data that cover a wide range of ontology languages, features and reference matches of the tracks. The alignments produced by extended Falcon-AO with those of Falcon-AO were compared. The evaluation is based on a standard measure applied by OAEI: precision and time as indicators. The evaluation is done to answer the following questions: How well and how long does extended Falcon-AO takes compared to Falcon-AO?

A. Precision

Is a value in the range [0, 1]; the higher the value, the fewer the wrong mappings computed. The precision measure could also be defined as follows:

A. Time

Is the value of time taken for the matching process after input of the two ontologies. For the running time, numerous experiments were conducted to show the impact of the proposed framework on the overall performance with no other applications running but the proposed system.

Experimental setup

The tests were performed on an Intel Celeron (R) 2.20 GHz Processor, 3.00 GB of RAM and 250 GB Hard Disk. The system has a Windows 7 Home premium, the application was

built on NetBeans IDE 7.2, WordNet as the lexical database, Java API for WordNet Search and Jena API.

V. Results and Discussion

Table 1 shows the alignment results of matching ontologies in the evaluation experiments.

Table 1: Results of the Experiments

Input			Falcon-AO				Extended Falcon-AO			
Sn	**Ontology $_1$**	**Ontology $_2$**	**Time (seconds)**	**Found**	**Correct**	**Precision**	**Time (Seconds)**	**Found**	**Correct**	**Precision**
1	Confof	IASTED	5	11	4	0.36363637	9	14	7	0.5
2	Cmt	Conference	4	18	9	0.5	8	19	10	0.5263158
3	Confof	Ekaw	5	21	13	0.61904764	9	23	15	0.65217394
4	Ekaw	IASTED	6	13	7	0.53846157	9	14	8	0.5714286
5	Edas	IASTED	8	11	7	0. 6363636	14	13	8	0.61538464
6	Cmt	Confof	4	11	6	0.54545456	7	11	6	0.54545456

The inputs are six pairs of ontologies from the conference track ontologies for which reference alignments are available. For each pair of the input ontologies, there is a total number of found alignments denoted **in Table 1** as **found**, correct found alignments denoted as **correct**, **time** in seconds denotes the time of execution after input of the two ontologies and the value of **precision** by extended Falcon-AO and Falcon-AO are also shown. The **found** and **correct** values are used to find the value of **precision**.

The result demonstrated that extended Falcon-AO showed better or similar precision. This can be explained from the visual observation of the values of the correct found alignment (correct) from Table1. Precision for the pairs of ontology 1-5 in Table1 increased, using the first pair, **Confof** and **IASTED** showed increase in total number of found alignment, correct found alignments and precision by 27.3%, 75% and 36.1% respectively. This shows that there is an increase in the number of correct found alignments and total found alignment thereby showing an increase in precision of the extended Falcon-AO system. It also showed an increase in execution time of up to 80%.

On the other hand the sixth pair **Cmt** and **Confof** showed an increase in percentage of values of total number of found alignment, correct found alignments, precision and execution time as 0, 0, 0 and 75 respectively. This indicates that even though no new matches were found by extended Falcon-AO, the execution time increased due to additional computation in extended Falcon-AO thereby showing an increase in time with no increase in precision. In general, this shows that there is a tradeoff between time and precision.

VI. Summary and Future work

The paper presented an approach of ontology matching based on Falcon-AO that provides an enrichment step to extend correspondences determined with standard match approaches. WordNet was exploited to determine the semantic type of correspondences between ontologies using Falcon-AO Ontology alignment system as the base system.

The approach delivered good results in the real benchmark ontologies used from the Ontology Alignment Evaluation Initiative (OAEI) in the conference Ontology track. The experiments carried out demonstrated that the proposed method resulted in an average improvement. As future work, WordNet can be used to detect initially false detected correspondences, e.g., by taking antonyms. Although this step will not add semantics to the mapping, it is potentially able to increase its precision.

References

[1] D.Jérôme, G. Fabrice and H.Briand. " Association rule ontology matching approach," International Journal on Semantic Web and Information Systems.vol 34 , 2007.

[2] E.Jérôme. An API for ontology alignment. International semantic web conference (ISWC) Montbonnot, pp. 698-712, 2004.

[3] E.Jerome and P. Shvaiko. A Survey of Schema-Based Matching Approaches. Journal on Data Semantics , pp.146-171, 2005.

[4] E.Jerome and P. Shvaiko. Ontology Matching: State of the art and future challenges. IEEE transactions on knowledge and data engineering , vol 21(1), pp.1-19, 2013.

[5] G.Stoilos, G.Stamou and S.Kollias. A string Metric for ontolog Alignment. Proceedings of the 4th International Semantic web Conference ,pp. 623-637, 2005.

[6] H.Faycal, S. Brigitte , N.B.Niraula and C.Reynaud. TaxoMap in the OAEI 2009 alignment contest. Proceedings of the ISWC'09 Workshop on Ontology Matching , 2009.

[7] H.Wei. Falcon-AO. Retrieved July 10, 2013, from Falcon-AO: ws.nju.edu.cn/Falcon-ao/,September , 2010.

[8] H.Zhang, W.Hu, and Y.QU . VDoc+: a virtual document based approach for matching large ontologies using MapReduce. Journal of Zhejiang University-SCIENCE C (Computers & Electronics) ,pp. 257-267, 2012.

[9] K.Saruladha, G.Aghila and B.Sathiya. A Comparative Analysis of Ontology and Schema Matching Systems. International Journal of Computer Applications , vol 34 (8), pp14-21, 2011.

[10] M.H.Seddiqui and M. Aono."Alignment Results of Anchor-Flood Algorithm". Proceedings of Ontology Matching Workshop of the 8th International Semantic Web Conference, Chantilly, VA, USA , 2009.

[11] M.M.Hussain and S. Srivatsa. "A study of Different Ontology Matching System". International Journal of Computer Applications, vol 37(8), pp. 10-16, 2012.

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

[13] W.Hu, Qu, Y., & Cheng, G. Matching large ontologies: A divide-and-conquer approach. Data & Knowledge Engineering , vol 67 ,pp. 140- 160, 2008.

[14] Y. Qu, W. Hu and G.Cheng . Constructing Virtual Documents for Ontology Matching. Proceedings of the 15th International World Wide Web Conference , pp. 23-31, 2006.

[15] Y.R.Jean-Mary, & Kabuka, M. R. (2009). "Ontology matching with semantic verification," Journalof Web Semantics , pp.235-251, 2009.

[16] Y.Wang, Liu, W., & A.D Bell. "A Structure-based Similarity Spreading Approach for Ontology Matching," pp.361-374, 2010.