Instance Based Matching System for Nuclear Ontologies

Pushpakumar R
PG Scholar, M.Tech - Software Engineering, Indian Institute of Information Technology, Srirangam, Tiruchirappalli, India.

Dr. M.Sai Baba
Scientific Information Resource Division, Resource Management Group, Indira Gandhi Centre for Atomic Research, Kalpakkam,Tamilnadu, India.

Dr.N.Madurai Meenachi
Scientific Information Resource Division, Resource Management Group, Indira Gandhi Centre for Atomic Research, Kalpakkam,Tamilnadu, India.

Balasubramanian P
Faculty, Department of Computer Science and Engineering, Indian Institute of Information Technology, Srirangam, Tiruchirappalli, India.

Abstract – Ontology matching is method interoperability viewer through heterogeneous data resources in the semantic web for mixing data semantically. Instance matching system is one of the important ontology matching systems, in many domains, such as the nuclear, ontologies are becoming gradually large thus presenting new challenges. We have developed a new instance matching system, focused on computational effectiveness and designed to switch ontologies, while maintaining most of the flexibility and extensibility of the Instance matching system. We calculated the efficiency of instance matching system two nuclear Ontologies, obtaining excellent run time results. Instance matching is the best system as measured in positions of F-measure. Similarly in terms of F-measure, instance matching system is good with the best performers in two of the task track that match whole ontologies.

Index Terms – Ontology Instance matching, Alignment, Automatch, Relationship map, word matcher, String matcher.

1. INTRODUCTION

Conceptualization has become the support to enable the fulfillment of the Semantic Web vision. Current days, ontology alignment has been taken as a key technology to answer interoperability problems through heterogeneous data sources. It takes ontologies as input and controls as output an alignment that is a set of messages among the semantically related entities of those ontologies. These communications can be used for various tasks such as ontology merging, data translation, query answering or steering on the web of data. Thus, matching ontologies allows the information and data expressed in the matched ontologies to interoperate. Though, success of the vision of Semantic Web depends on the accessibility of semantic connected data. Semantic related data are all about associated data for example, people, places and things that are joined to respectively other. Each identifiable information known, as instance. This research can just be finish by hand for ontologies [1].

Ontology instance matching equates different characters within same or heterogeneous ontologies with the goal of recognizing the same real world things. It also describes the degree of semantic relationship among each other [2]. The ontology instance matching problem has been broadly studied in several application domains where it is known with different names such as uniqueness recognition, record linkage, and entity steadfastness problem and so on according to the requirements that need to be fulfilled.

It is imperative to notice that ontology and instance matching are similar to the idea of database schema matching and record relation in the research domain of record. Instance matching plays a critical role in semantic data integration as it interconnects all the islands of instances of semantic world to achieve the interoperability and information incorporation issues. Ontology instance matching is alike important in ontology population as it, helps to properly perform the insertion and update operation and to discover relationship among the new received instance and the set of instance before stored in the ontology.

Given the growing importance of matching very large ontologies, mainly in the nuclear domain, we have developed a novel instance matching, derived from instance matching system and attentive on the effective matching of very large ontologies, the instance matching system framework. Unlike

instance matching system it is not designed to support user communication, but it keeps the flexibility and extensibility of the original framework.

In this paper, we present the instance matching system and its evaluation on the Anatomy and nuclear instance. This paper is ordered as follows: Section 2 presents instance matching problems, Section 3 presents the instance ontology matching methods in depth and Architecture overview, Section 5 presents and discusses the results of the evaluation, and Section 6 presents the conclusions of our work.

2. INSTANCE MATCHING PROBLEMS

Instance matching to find out nearness between two individuals referred to the same real world object an instance matching algorithm essential to solve the problem of different kinds of heterogeneity such as value transformation, structural heterogeneity in addition logical heterogeneity.

Value Transformation: Instances hold lexical information as their property values that may contain error or be denoted using different standard format, such as dates or person name in different countries. This concern has been addressed in the field of record linkage research.

Structural heterogeneity: Different record linkage, schema and instance are more firmly related in instance matching. On behalf of an instance lexical information is often connected with a property by direct order of characters or by other instance stately different levels of depth in property representation. Unlike aggregation standards of properties, like full name as signified by first name and last name together, tempts extra level of troubles in instance matching. Additionally missing values of properties and several values of a single property through knowledge base introduce heterogeneity to represent same actual world instance contrarily.

Logical heterogeneity: There are amount of features prompting logical heterogeneity in instance representation. Equal instance can be instantiated into different subclasses of the same class or into more general classes without altering the meaning. "Hanif", presence a person, can be definite by a subclass man without changing its meaning. Furthermore implied value specification and implicit similar classes defined by restrictions also introduce heterogeneity in defining similar instances. However instance defined by split classes are ensuring different meaning even if they are holding similar type of descriptions.

3. ONTOLOGY MATCHING METHODS

The method of matching or aligning two input ontologies (one source ontology and one target ontology) contains in finding semantic associations between the classes of the source ontology and the classes of the target ontology. In the setting of this paper, these semantic relationships are controlled to correspondence relationships, and are named mappings. The set of mappings among two ontologies is called an alignment [4]. Ontology instance matching systems use matching algorithms, named matchers, which allocate a mathematical value to each mapping. This mathematical value reproduces the semantic similarity among terms. These matchers can function at different levels, with the element level and the basic level [7].

Element level matchers investigate concepts or their instances in isolation, ignoring their dealings with other concepts or instances. These matchers can use internal knowledge only, that is, information controlled in the ontology itself, or integrate external knowledge in the form of reusable alignments, upper or domain ontologies, and other language resources. A general inner knowledge element level matching technique is based on the lexical matching of the labels related with ontology concepts.

Structure level methods compare ontology ideas or their instances based on their relationships with other concepts or instances. They can also use outside knowledge, such as instances that are not part of the ontology or prior alignments. Homogeneous in which the unlike kinds of data are processed by suitable matchers. Heterogeneous in which the same input, is used by separate matchers. Some methods use this concept to spread similarities through the ontology [10] [8].

Later the similarities among ontology concepts have been computed, it is essential to use a global approach to arrive at a final optimized alignment. These methods can include trimming, which applies thresholds to ensure only the best matches are considered, or best weight matching, which improve the global similarity [5].

4. ARCHITECTURE OF INSTANCE MATCHING SYSTEM

Overview

The instance matching system was programmed in Java and developed in NetBeans. The core framework includes four modules: the ontology loading module, the ontology matching module, filtering module, similarity module. The ontology loading module is responsible for loading the input ontology files and creating ontology objects (Figure 1). It is also responsible for loading external ontologies used as background knowledge. The instance matching module is responsible, for aligning ontology objects by joining one or more matching algorithms and one or more selection steps.

Like instance matching, the instance matching ontology matching module was designed with flexibility and extensibility in mind, and thus permits for the insertion of virtually any matching algorithm. The instance matching system also includes four key data structures: the instance

matching system, the Lexicon, the Relationship Map and the instance Alignment. The first two data structures are the main element of ontology objects for example demonstration of ontologies use in to support the matching process.

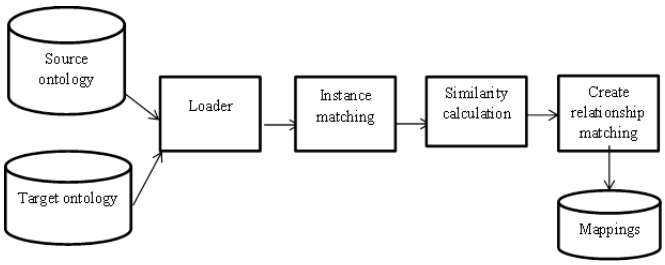

Fig.1 Instance matching system Architecture

As their names mention, the Lexicon stores the lexical information of an ontology and the Relationship Map stores the structural information of an ontology. The Alignment stores the set of mappings among two ontologies produced by one or more instance matching algorithms.

Ontology loading process

Such as is the case in instance matching system, the instance matching ontology loading is currently based on the jena2 ontology API, and the first step in the loading process is to read the ontology file into memory as a jena ontology Model. The change among the two system is that instance matching stores in internal structures all the information from the Ontology model that is essential for ontology matching, while instance matching system retains the ontology model in memory through the matching process. Afterward using jena to read an input ontology file, the instance matching ontology loading module abstracts from the ontology model.

The following ontology related organized under an ontology object the URI or the ontology. All list of URI of all named classes in the ontology that go to the ontology namespace. All list of local class and names. A lexicon that holds the local names of all listed classes their labels, and all their synonyms. A Relationship map that as a contain part of relationship between the listed classes. Structure the relationship map is optional since the relationship are unnecessary for many matching algorithms and this step takes approximately sixty percent of the total ontology loading time.

Ontology Instance Matching: Easily, Ontology instance matching, as well defined before, compares different individuals within same or heterogeneous ontologies with the goal of finding the same real world objects. Supposed two sets S (source) and T (target) of instances inside same ontology or through different ontologies, a semantic similarity measure $\sigma:s*t\rightarrow[0,1]$ and a threshold $\&\in [0,1]$, the goal of instance matching assignment is to compute the set $m=\{(s,t), \sigma(s,t) \geq \&\}$ [2]. The instances of S are linked to the instances of T and if their attraction is greater than a threshold value then the instance brace is measured as a member of set M and M contains all aligned instance pairs.

Lexicon: A new part that was combined in the instance matching system lexicon was a system of weight to replicate the consistency of each source. For instance synonyms gained from has precise synonym statement are in code more consistent than synonyms found from has related synonym statements as they should be nearer in meaning to the concept labeled by a class. The inner structure of the lexicon contains on two multi maps covering classes, names and attributions, with one needing the class as key and the other requiring the name as key. Therefore the lexicon can be asked by both class and name at effectively no computational cost.

Relationship Map: The relationship map stores, all is and part of paths in an ontology with transitive closure, and contains the distance of each path in number of edges. It also stores all direct disjoint clauses in an ontology. Like the lexicon the relationship map is based on multi maps. It contains two multi maps for relationship which contain ancestors, descendent and relationship, with one having the ancestor as key and the other one having the descendent as key. It also includes a Hash Map of sets for disjoint clauses, linking each class to all classes that are disjoint with it. Hence the connection map can be queried to find all descendent of a class, all ancestors of class and all classes disjoint with a class at virtually no computational cost.

Alignment: The internal structure of alignment in instance matching system is identical to that of instance matching. It includes two multi Maps that contain the source class target class and similarity with one having the source class as key and the other one having the target class as key. This enables effective enquiring of mapping by class and means that alignment links to a spare matrix. In addition, alignment also contains a list structure that enables storing and thus simplifies selection.

Automatch: Additional system using instance-based matching methods is Automatch [5]. For each domain experts create attribute dictionaries holding all probable values and the fitting probabilities of existence. The instances of an ontology are matched against these dictionaries, and the resulting mapping offers the basis for calculating an individual match score for each attribute. The some match scores are summed up and lastly a mapping is determined by applying a minimum cost maximum flow graph algorithm.

Similarity matching method

We executed two matchers the word matcher and string matcher. The first were implemented as primary matchers and second one was implemented secondary matcher.

Word matcher is a word based string similarity algorithm that measures the similarity between two classes through a weighted jaccard index between the words present in their name. it is derived from vector based multi word matcher of ontology matching, but is based on lexical cross search whereas the latter requires an all against all comparison. The primary step in the Word Matcher is the origin of a Word Lexicon from the Lexicon of each ontology and calculating the frequency and evidence content of each word. The evidence of each word is assumed by the inverse logarithm of its frequency [3].

String Matcher is a string similarity algorithm that implements a variation of similarity metrics and was straight ported from instance matching. Then it makes non-literal string contrasts, it involves an all-against-all assessment of the ontologies and then is a secondary matcher in instance system.

5. EXPERIMENT AND EVALUATION

We experimented with the datasets of nuclear ontology and instances take on instance matching system. The datasets contain instance of nuclear reactors and nuclear power plant there are more number of class and properties values as took to test case. The data set has probably to test based on the new proposed system. Eventually the system has given most appropriate related matching result in take test cases. Instance matching system was used in 0.6 similarity threshold for string and word matcher. The instance matching system capable to match less time target ontology and Source ontology was load few second then show result 0.98 of probability relative values. Proposed system advantage in terms of run time, we assume to get results similar to those of the best other systems in these jobs, whereas taking less time.

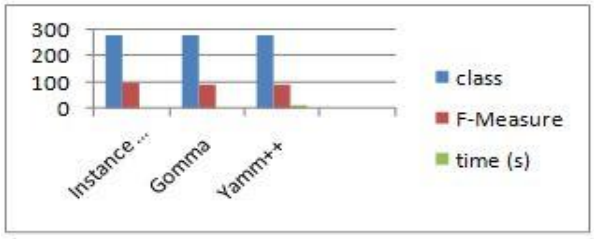

Fig 5.1 System Performance

6. CONCLUSION

Complete, the instance matching system has satisfied the goals for which it was designed. On the one hand, it was positive to match very large ontologies professionally, in a modest time when compared with top ontology matching systems. On the other hand, it was able to produce high quality results in the Structure track, which beaten even those of previous instance matching. In comparison with instance matching, instance matching system represents a considerable progress in terms of competence without losing performance, as measured in terms of precision, recall, and F-measure. Additionally, instance matching system shares the focus of ontology matching on flexibility and extensibility, which are part of its design strengths [4].

Instance matching system can increase even further upon these results, since that scalability to large and large ontologies has been, until now, the main focus of the work behind instance matching system. Concerning the Ontologies track, instance matching obtained outstanding run times, but it is clear that performance can be upgraded specially recall once we increase our focus.

In point, even without fixing on performance, the results obtained by instance matching system. Thus, we expect instance matching system to place between the very best systems in these tasks, with the occurrence of appropriate external data and a more whole matching configuration. As We assume to circumvent this problem by using a word based matching algorithm, which together with the inclusion of external data, will enable instance matching system to obtain good results in this task while remaining between the fastest systems. Additional developments to the instance matching system will contain advanced strategies for using external resources, repairing alignments, and using semantic similarity in the context of structural matching [8].

REFERENCES

[1] Katrin Zaiß and Tim Schluter and Stefan Conrad. “Instance-Based Ontology Matching Using Different Kinds of Formalisms” World Academy of Science, Engineering and Technology Vol:3 2009.

[2] R Pratap, D Nath, H Seddiqui, M Aono,” An Efficient and Scalable Approach for Ontology Instance Matching” Journal of Computers, Vol 9, No 8 (2014), 1755-1768, Aug 2014.

[3] Couto, F., Silva, M., Coutinho, P.: Finding genomic ontology terms in text using evidence content. BMC Bioinformatics 6(suppl. 1), S21 (2005)

[4] Cruz, I.F., Palandri Antonelli, F., Stroe, C.: AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. PVLDB 2(2), 1586–1589 (2009).

[5] Cruz, I.F., Palandri Antonelli, F., Stroe, C.: Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 551, pp. 49–60 (2009).

[6] J. Berlin and A. Motro, “Database Schema Matching Using Machine Learning with Feature Selection,” in Advanced Information Systems Engineering, 14th International Conference, CAiSE 2002, Toronto, Canada, May 27-31, 2002, Proceedings, 2002, pp. 452–466.

[7] Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag New York Inc. (2007).

[8] Pesquita, C., Stroe, C., Cruz, I.F., Couto, F.: BLOOMS on AgreementMaker: Results for OAEI 2010. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 689, pp. 135–141 (2010).

[9] S. Auer, J.Lehnann and A. C. N. Ngomo,”Introduction to linked data and its life cycle on the wed,” Reasoning Web 2011, pp. 1-75.

[10] Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: Proceedings of the 18th International Conference on Data Engineering, pp. 117–128 (2001).