Light-Weight Ontology Alignment using Best-Match Clone Detection

Paul L. Geesaman
School of Computing
Queen's University
Kingston, Ontario
Email: geesaman@cs.queensu.ca

James R. Cordy
School of Computing
Queen's University
Kingston, Ontario
Email: cordy@cs.queensu.ca

Amal Zouaq
Department of Mathematics
and Computer Science
Royal Military College of Canada
Kingston, Ontario
Email: amal.zouaq@rmc.ca

***Abstract*—Ontologies are a key component of the Semantic Web, providing a common basis for representing and exchanging domain meaning in web documents and resources. Ontology alignment is the problem of relating the elements of two formal ontologies for a semantic domain, in order to identify common concepts and relationships represented using different terminology or language, and thus allow meaningful communication and exchange of documents and resources represented using different ontologies for the same domain. Many algorithms have been proposed for ontology alignment, each with their own strengths and weaknesses. The problem is in many ways similar to near-miss clone detection: while much of the description of concepts in two ontologies may be similar, there can be differences in structure or vocabulary that make similarity detection challenging. Based on our previous work extending clone detection to modelling languages such as WSDL using contextualization, in this work we apply near-miss clone detection to the problem of ontology alignment, and use the new notion of "best-match" clone detection to achieve results similar to many existing ontology alignment algorithms when applied to standard benchmarks.**

***Index Terms*—Clone detection techniques; OWL; ontology matching; ontology alignment**

I. Introduction

Ontologies, and more specifically domain ontologies, are knowledge representations that describe the semantics of a real-world conceptual domain in a way that is understandable by a computer. The purpose of these ontologies is the sharing and reuse of domain knowledge between humans and machines. As such, ontologies represent the backbone of the Semantic Web.

The W3C consortium has recommended the Web Ontology Language (OWL) as one of the languages to represent ontologies for the Semantic Web. OWL is a formal, RDF/XML-based language explicitly designed for expressing domain ontologies. OWL ontologies consist of concepts, links between concepts, and other information such as properties and instances of concepts.

Although the aim of the Semantic Web is to promote reuse and sharing, various ontologies about the same domain can be created by different communities of experts, thus leading to problems in communication and hindering the sharing of documents and knowledge. A challenge for the Semantic Web community is bridging the gap between these different ontologies by determining their similarities. The overall goal is to allow for the cross-referencing of ontologies and their components within the larger system of the Semantic Web.

Ontology alignment is the problem of identifying the concepts of a target ontology that represent the same, or a subset of, a corresponding concept in a given source ontology. It includes identifying the correspondence between the RDF/XML elements which make up an OWL ontology (classes, properties, individuals, and so on) in a source ontology to the elements in a target ontology. Ontology alignment is a difficult problem, and alignment algorithms can be expensive in time and space.

In this work we propose applying near-miss cross-clone detection to the ontology alignment problem, in order to examine and overcome some of the problems faced by other ontology alignment algorithms, in particular running time. We analyze OWL RDF/XML source code for near-miss cross-clones between the source and target ontologies, using the clone relationship to find corresponding ontology elements. Near-miss (type 3) clone detection [1] is particularly useful in discovering copies of source code with differences such as small modifications in structure, parts of code removed or added, and names modified. These are the same kinds of changes that are typical of the differences between similar ontologies, which may contain small modifications in subclass structure, properties removed or added, class names modified or synonyms used.

In our previous work [2], we have applied near-miss clone detection to WSDL, another modelling language which, like OWL, is expressed in XML. Using a new technique called contextual clones, we introduced context to XML elements referring to other elements by replacing each reference with the element it refers to. The result is fully self-contained elements that explicate their entire definition and have no dependence on their environment, making them rich candidates for clone comparison.

In this paper we explore whether this same technique can be used in a new approach to ontology alignment. We begin by introducing the ontology alignment problem, concentrating on the representation of concepts in formal ontologies and the challenges of matching concepts in one ontology to those in another. We describe a process for contextualizing concepts expressed in RDF/XML by inlining reference elements that

```
<!-- A class -->
<owl:Class rdf:ID="Monograph">
  <rdfs:subClassOf rdf:resource="#Book" />
  <rdfs:subClassOf>
    <owl:Restriction>
  <owl:onProperty rdf:resource="#chapters" />
  <owl:allValuesFrom rdf:resource="#Chapter"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<!-- A property -->
<owl:ObjectProperty rdf:ID="chapters">
  <rdfs:domain rdf:resource="#Reference"/>
  <rdfs:range rdf:resource="#Chapter" />
</owl:ObjectProperty>

<!-- An instance -->
<Monograph rdf:about="#a108048723">
  <rdfs:label>Object-Oriented Data Modeling</rdfs:label>
  <publisher rdf:resource="#a85849488"/>
  <title>Object-Oriented Data Modeling</title>
  <date>
    <Date>
      <year rdf:datatype="&xsd;gYear">
        2000
      </year>
    </Date>
  </date>
</Monograph>
```

Fig. 1: Example excerpt from an OWL ontology

refer to other elements located elsewhere in the file to produce fully expanded standalone concept descriptions for all of the concepts in each of the two ontologies to be aligned. We then apply the NICAD clone detector [1] to detect near-miss cross-clones between the two sets of contextualized concept descriptions, and using the near-miss clones of a concept in the first ontology from the second ontology as proposed concept alignments. We tune this process by varying the near-miss threshold to optimize precision and recall with respect to the gold standard answers for a standard set of ontology alignment challenge problems. Finally, we show how to increase precision by considering only the best-match clones, those with the highest similarity, as the proposed alignments.

II. Ontology Alignment

In this section we introduce the characteristics of formal ontologies, the OWL language and the ontology alignment process.

A. Formal Ontologies

An OWL ontology includes descriptions of classes, properties and their instances. Classes describe concepts, the main building block of an ontology. They are generally organized in a hierarchy through the *subClassOf* statement. Classes have properties that describe them. Properties can be either object properties or data types properties. Properties may have a hierarchy through the *subPropertyOf* statement. Finally, classes can have instances, which are members of the classes.

Figure 1 shows an excerpt from an OWL ontology in the dataset introduced later in this paper, showing an example of a class, a property, and an instance in the OWL XML representation.

```
<map>
  <Cell>
    <entity1 rdf:resource="sourceOntology;MastersThesis"/>
    <entity2 rdf:resource="targetOntology;MScThesis"/>
    <measure rdf:datatype="XMLSchema;float">1.0</measure>
    <relation>=</relation>
  </Cell>
</map>
```

Fig. 2: Example ontology alignment

B. Alignment Process

The ontology alignment process is done with a source and target ontology. Both ontologies are given to the alignment tool, and a proposed alignment is provided as output. The tool's alignment can then be compared to a reference alignment for the sake of evaluation.

Figure 2 shows an example reference alignment for concepts from two similar ontologies, taken from the gold standard which we explain in greater detail in Section V-A. In this example, the concept *MastersThesis* from the source ontology should be aligned with *MScThesis* from the target ontology. The alignment has an exact correspondence between the entity in the source ontology and the entity in the target ontology, and they are equal. The alignment process may permit ”=”, ”$\sqsubseteq$”, and ”$\sqsupseteq$” relations; most of the benchmark dataset's alignments are ”=” relations.

III. Contextualizing Concepts

In order to experiment with best-match clone detection on the ontology alignment problem, we began our experimentation with alignment of OWL classes. Once we have implemented and tuned our alignment successfully for classes, we will apply the same process to properties and instances. In this paper, we describe our results for classes only, which represent 19% of the gold standard's entities.

A challenge in applying clone detection to the alignment problem was finding a way to meaningfully compare concept classes for clones. Using the notion of contextual clones from our previous work [2], we recursively inlined *subclassOf* references to other classes in each class definition. This means that for each concept class we build its entire concept tree, giving us a way to meaningfully compare entire classes to one another.

The most important piece of information used in this contextualization is the OWL *<rdfs:subclassOf rdf:resource=[resourceName]>* element, where *[resourceName]* refers to another element. Replacing the *subclassof* elements with their referred class elements is done recursively, resulting in a fully contextualized representation of each class in the OWL ontology.

Figure 3 shows an example of an OWL concept class element before and after contextualization.

IV. Concept Clones

In order to use near-miss clone detection on our contextualized concept classes, we created a NICAD plug-in to parse, contextualize and extract class elements from OWL ontology files. We began with a TXL [3] grammar to parse

```
<owl:Class rdf:ID="MastersThesis">
  <rdfs:subClassOf rdf:resource="#Academic" />
</owl:Class>
```

(a) Original OWL concept definition

```
<owl:Class  rdf:ID="MastersThesis">
  <rdfs:subClassOf>
    <owl:Class  rdf:ID="Academic">
      <rdfs:subClassOf>
        <owl:Class  rdf:ID="Reference">
          <rdfs:subClassOf>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#date"/>
              <owl:maxCardinality rdf:datatype=
                "&xsd;nonNegativeInteger">
                1
              </owl:maxCardinality>
            </owl:Restriction>
          </rdfs:subClassOf>
          <rdfs:subClassOf>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#title"/>
              <owl:maxCardinality rdf:datatype=
                "&xsd;nonNegativeInteger">
                1
              </owl:maxCardinality>
            </owl:Restriction>
          </rdfs:subClassOf>
        </owl:Class>
      </rdfs:subClassOf>
    </owl:Class>
  </rdfs:subClassOf>
</source>
```

(b) The same concept definition after contextualization

Fig. 3: Example OWL concept contextualization

OWL, adapted from the W3C OWL specification [4], and used a TXL transformation to implement reference inlining to contextualize and extract OWL concept classes in a way analogous to the contextualization and extraction of WSDL operations in our previous work [2]. Like the WSDL extractor, our NICAD plug-in for OWL also performs some minor normalization, eliminating irrelevant elements from the XML representation.

Once we had extracted the contextualized classes from a source and target ontology, we used NICAD to check for cross-clones between them. We began using NICAD's default settings, simply looking for near-miss cross-clones at the 30% near-miss threshold, and using the target ontology cross-clone relationship as the proposed alignments for each concept in the source ontology. Even in this naive default setting, some alignments were found and many tests had good recall by comparison with the gold standard, but there was far too much noise and precision was very low.

Part of the problem was that in many cases there were many good near-miss cross-clones for a concept in the source ontology. While we were hoping that the near-miss clone relationship would identify only the closest matches below the threshold, in practice there were often many of these.

By examining what was happening when an ontology was compared to itself, we realized that the match deemed correct by the gold standard was typically the cross-clone with the highest similarity (the "best match"), and that other clones found at the NICAD threshold were not as interesting. We also noticed that, by using only this highest similarity, or *best-match* definition, most concepts in the source ontology would yield a single match, which is ideally what we were seeking.

To implement this new best-match clone definition, we modified our process to filter NICAD results to yield only those cross-clone pairs that match at the highest similarity value, returning a single alignment for each concept in the best case, or a set of best-match alignments in the general case.

V. An Experiment

In order to test the efficacy of near-miss clone detection and our new best-match clone definition in practice, we designed an experiment to compare our results against an accepted benchmark at various NICAD parameter settings, in order to determine the best clone detection settings for the ontology alignment problem. While good settings for near-miss code clone detection have been explored in other work, there was no reason to believe that those would be best for this very different problem.

A. The OAEI Benchmarks

The Ontology Alignment Evaluation Initiative (OAEI) is an organization that provides datasets for evaluation of ontology alignment tools. Its goal is to improve ontology alignment techniques, and the evaluation of the techniques at conferences [5]. Since our new lightweight method is designed to compete with other ontology alignment tools, the OAEI benchmarks obviously make good examples with which to test it.

The dataset we chose for our experiment is the *biblio* dataset that was used in the OAEI's 2011 ontology alignment challenge [6]. This dataset is designed to be a comprehensive test for alignment tools, and because the 2011 competition results have been published, we could compare our results to other state-of-the-art tools. We chose the 2011 dataset before the 2012 alignment challenge results were available, and we plan to apply our technique to the 2012 dataset once we have optimized our tool.

The 2011 *biblio* dataset defines 173 entities to be matched, including 33 named classes, 24 object properties, 40 data properties, 56 named individuals, and 20 anonymous individuals. The source ontology is compared to a set of target ontologies, each of which was obtained by applying transformations to the source ontology. Every problem in the challenge has a defined gold standard alignment that can be used to evaluate the results obtained by an alignment tool.

The dataset is divided into three levels: the 100-level, 200-level, and 300-level. The 100-level tests are simple tests that have minor changes to the ontology - for example presenting the same labels in OWL Lite, a light-weight version of OWL which is a subset of OWL's syntax.

The 200-level tests have a number of different transformations performed on the data in a systematic way. The tests stack with one-another to create more difficult tests. The transformations are: entity names can be replaced with random strings, synonyms, or names are translated to different languages; classes can be expanded, or have restrictions

TABLE I: Measurements of traditional clone detection with different thresholds

	Traditional		
Threshold	**Precision**	**Recall**	**F-measure**
.00	1.00	0.03	0.06
.10	0.98	0.04	0.08
.20	0.88	0.19	0.31
.30	**0.69**	**0.42**	**0.52**
.40	0.51	0.47	0.49
.50	0.46	0.50	0.48
.60	0.44	0.51	0.47
.70	0.42	0.52	0.46
.80	0.17	0.56	0.26
.90	0.17	0.68	0.27

imposed on the classes discarded; comments are suppressed or changed to a different language; the hierarchy of the ontology is suppressed, expanded, or flattened; instances are suppressed; properties are suppressed, or have restrictions imposed on them discarded.

For this first experiment, we are analyzing only classes, so transformations that affect classes, hierarchy, and naming have a greater impact on our matching results than transformations on properties, instances, and comments (although these can form part of the contextualized classes).

The 300-level tests are real-life ontologies describing the same domain, but the OAEI webpage for the 2011 challenge says that the gold standards for these ontologies contain flaws, and thus the OAEI did not publish the results of tools for the 300-level tests in the 2011 alignment challenge. For this reason we have not included these in our experiment.

B. Tuning Precision and Recall

To evaluate our results, we use the standard information retrieval metrics of the ontology alignment community, precision, recall and f-measure, to compare our result alignments with the gold standard for each problem.

Precision is a measure of how much confidence one may put in the alignment returned by a tool.

$$precision = \frac{|\{Clone\ Pairs\} \cap \{Gold\ Standard\ Pairs\}|}{|\{Clone\ Pairs\}|}$$

Recall is a measure of how well a tool can discover the gold standard alignments.

$$recall = \frac{|\{Clone\ Pairs\} \cap \{Gold\ Standard\ Pairs\}|}{|\{Gold\ Standard\ Pairs\}|}$$

Finally, F-measure is the harmonic mean of precision and recall. F-measure is the metric we have chosen as the primary measure of our new method's success, and thus the one we want to maximize. If two NICAD thresholds have the same F-measure, then the lower threshold will be considered better since it typically has a lower NICAD computation time.

$$F - measure = \frac{2 * precision * recall}{precision + recall}$$

TABLE II: Measurements of best-match clone detection with different thresholds

	Best-match		
Threshold	**Precision**	**Recall**	**F-measure**
.00	1.00	0.03	0.06
.10	0.99	0.04	0.08
.20	0.95	0.19	0.32
.30	0.88	0.41	0.56
.40	**0.80**	**0.45**	**0.58**
.50	0.79	0.45	0.57
.60	0.79	0.45	0.57
.70	0.77	0.45	0.58
.80	0.52	0.46	0.49
.90	0.53	0.51	0.52

C. Traditional Clone Detection

In our first experiment, we used NICAD's default cross-clone algorithm at various near-miss thresholds, using the set of all target ontology cross-clone pairs for a contextualized concept class of the source ontology as the proposed alignment. Rather than use NICAD's clone classes, we treat NICAD's raw output of clone pairs as proposed "=" alignments with correspondence 1.0 . To evaluate these results, we treated every alignment in the gold standard as a reference pairing, and compared the sets of clone pairs returned by NICAD for each source concept class to the reference pairing in the gold standard.

Our running time for 110 tests and measurements was approximately 30 seconds on a machine with a 2.2 GHz Intel Core i7 processor and 4 GB of RAM running MacOS X version 10.7.5. Our results are displayed in Figure 4a and Table I.

Using this raw application of NICAD clone detection, our best results are obtained at threshold 0.30 (emboldened in Table I), where we have obtained the highest F-measure. Based on this finding, we believe that the best threshold for using clone-detection for the task of ontology alignment may be near this value.

D. Best-Match Clone Detection

We repeated our experiment using our new notion of "best-match" clone detection. Best-match clone detection applies a filter to the results of traditional clone detection. For each contextualized concept class in the source ontology, we find the clone pair(s) in the target ontology that have the highest similarity and filter out the rest. If two or more clone pairs share the highest similarity, then all of that similarity are kept.

Once again we assume that each best-match clone pair returned by the tool is a proposed alignment, and we evaluate our tool using the same measurements and conditions as in Section V-C. Table II and Figure 4b show the results for our best-match clone method. For the best-match algorithm, we conclude that the best threshold for our dataset is 0.40 because it returns the highest F-measure, and takes less time to compute than threshold 0.80, which has the same F-measure.

NICAD with best-match filtering was able to complete the entire 110 tests of the biblio benchmark in approximately 50 seconds running on a machine with a 2.2 Intel Core i7 processor and 4 GB of RAM running MacOS X version 10.7.5, still much faster than other algorithms.

Even with best-match clone detection, the algorithm may return a set of possible alignments (clone pairs) for each source concept class. Once again evaluation is calculated using precision, recall, and F-measure compared to the gold standard. Our best-match results are shown in Table II and Figure 4. As we can see in Table II by comparison with Table I, the best-match filtering at threshold 0.4 gives a significant improvement over raw cross-clone detection at its best threshold (0.3) in precision, and improves the F-measure and recall as well.

E. Limitations

Thus far we have attempted to match only concept classes, and have not yet worked on aligning properties or instances. However, those elements are not fundamentally different, and we are confident that the same contextualization strategy and best-match filtering will serve for them also.

While our results so far seem promising, there are alignment problems in the OAEI challenge set that have characteristics that pure near-miss clone detection cannot solve, such as randomly renamed concept classes, that have limited our success. As we add consistent renaming to our clone detection normalization, we expect to see better results as renamed concepts will be more accurately matched.

In our work so far we have also have suppressed comments, but including them may be beneficial, as some tests expect comments to provide hints for alignment.

VI. Related Work

Many other ontology alignment tools have been used in past years of the ontology alignment evaluation challenge. The methods described below are from the description of a recent survey paper have listed tools which have made entries in alignment competitions for several competitions [7].

For purposes of evaluation, we present our results on the biblio dataset in the context of other tools which were entered in the 2011 competition [8] in Figure 5 and Table III. While our lightweight clone-based methods are by no means at the top of the list, as we can see our best-match method already fits well into the table of results for much more mature tools.

In terms of performance, in the competition the tools were run on a 3 GHz Xeon5472 quad-core processor with 8GB of RAM running Linux Fedora 8. In this configuration, the other tools had running times between 1.07 minutes and 28.94 hours [9]. Although our results are not necessarily directly comparable, even in its present prototype stage our clone-based tool took less time on a less powerful machine than any of the other tools, and we expect our running time to improve as we do more normalization and tuning.

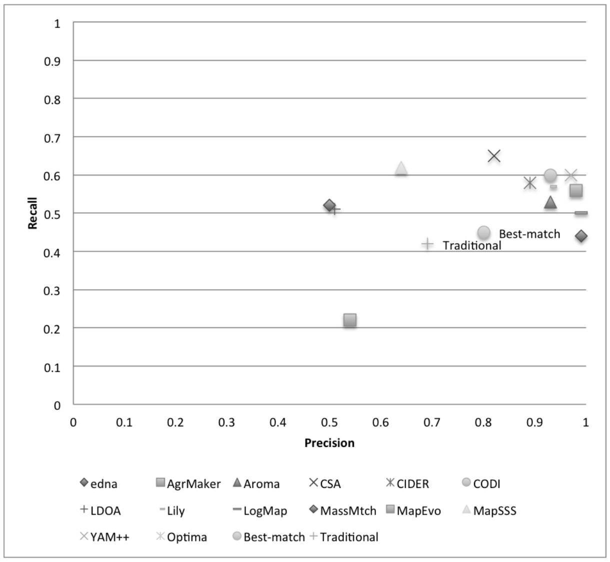


Fig. 5: Alignment tool results on the biblio dataset, 2011

While our method is knowledge- and domain-independent, many other tools take advantage of external background knowledge or are created for specific domains [10], [11], [12], [13], [14], [15], [16], [17]. One such source of background knowledge is WordNet [18]: a lexical database of nouns, verbs, adverbs, and adjectives. The database can be queried for synonyms, antonyms, and other relationships between words in the English language. Another widely-used source of knowledge is DBpedia [19]. DBpedia is a database of information extracted from Wikipedia. A user can query the information contained within DBpedia, and they can link other data with the database. As opposed to other methods, our tool is naive and domain-independent.

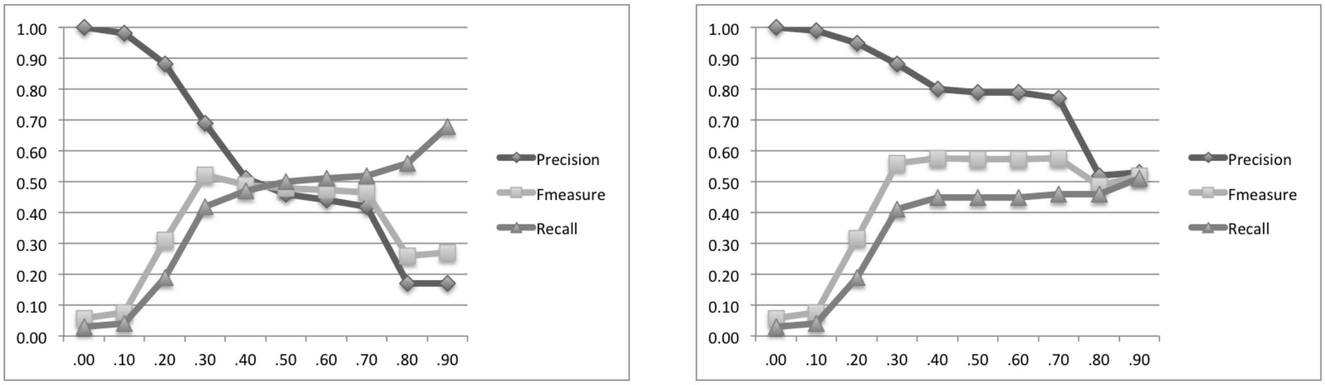
Some alignment tools treat the problem as a graph problem, and examine the structure of the graph [20], [21]. Lily [22] is one such tool which examines subgraphs in order to make alignments. In a similar fashion, LogMap uses logic and structure in order to generate an alignment [23].

Some tools use rules exploiting OWL characteristics to do alignment [24]. Similarly, naive Bayes classifiers have been used to align ontologies [10], [14]. Other approaches use probabilities with rules to generate alignments [25].

CIDER [8], one of the tools reported in the 2011 competition uses rules of inference and background knowledge in order to extract features of an entity. The features are then used as an input to an artificial neural network, which outputs a matrix of similarity between entities. The alignment of the ontology is based on the similarity matrix.

Term frequency by inverse document frequency (TF · IDF) is used by AgreementMaker [14] and other information retrieval methods [26]. TF · IDF makes a table of terms by documents, and each entry presents how many times the term appears in the document multiplied by the inverse of how many time the term appears in all the documents. The cosines of elements are calculated to generate a similarity metric which is then used to aid in alignment.

Falcon [27] is an ontology matching algorithm that has the same basic steps as our tool: the tool partitions ontologies to



(a) Traditional Near-miss Clone Detection

(b) Results After Best-match Filtering

Fig. 4: NICAD Results With and Without Best-match Filtering

TABLE III: Measurements of tools on the 2011 Biblio dataset in descending order. Thresholds selected are 0.30 and 0.40 for our traditional and best-match clone detection respectively

Tools sorted by Precision		**Tools sorted by Recall**		**Tools sorted by F-measure**	
LogMap	0.99	CSA	0.65	MapSSS	0.77
MassMatch	0.99	MapSSS	0.62	CODI	0.74
AgreementMaker	0.98	CODI	0.60	YAM++	0.74
YAM++	0.97	YAM++	0.60	CSA	0.73
Aroma	0.93	CIDER	0.58	AgreementMaker	0.71
CODI	0.93	Lily	0.57	CIDER	0.70
Lily	0.93	AgreementMaker	0.56	Lily	0.70
CIDER	0.89	Aroma	0.53	Aroma	0.68
CSA	0.89	Optima	0.53	LogMap	0.67
Best-Match Clone tool	**0.80**	edna	0.52	MassMatch	0.61
MapSSS	0.80	LDOA	0.51	**Best-match clone tool**	**0.58**
Traditional Clone Detection	**0.69**	LogMap	0.50	Optima	0.56
Optima	0.60	**Best-Match Clone tool**	**0.45**	**Traditional Clone Detection**	**0.52**
MapEvo	0.54	MassMatch	0.44	edna	0.51
LDOA	0.51	**Traditional Clone Detection**	**0.42**	LDOA	0.51
edna	0.50	MapEvo	0.22	MapEvo	0.32

smaller blocks, matches blocks, and discover blocks. The tool also uses the subClassOf relation to find relationships between classes, and to discover structural similarities. It's different in that it uses clustering algorithms to generate alignments. Other algorithms use clustering techniques as well [28].

Some tools use edit distance, and in some cases specifically Levenstein distance is used to find alignment [11]. Hertuda [29] tokenizes entities and examines the edit distance of the entities as strings using the Damerau-Levenstein algorithm with threshold selected separately for classes, properties, and instances. The tool returned the Hertuda was not included in the evaluation because its results are not comparable to the dataset we used. Hertuda was entered in the 2011.5, and 2012 competition, but not the 2011 competition.

MapEVO is an algorithm that uses evolutionary programming to make an ontology alignment [30].

Our tool works with a 1-to-many alignment, however some tools have been developed to do many-to-many relationships. We had decided to work primarily with 1-to-many for the sake of comparing two ontologies. We can examine in the future whether NICAD's clone classes may be of use in many-to-many matching. Our method only finds equalities, but some of the of the alignments propose a "<" alignment.

By comparison with other tools, our method is fast, and scales well to very large ontologies due to its basis on a production code clone detector. Unlike many of the other tools, it requires no training on the domain or problem set to be applied, and requires no external sources of domain knowledge. It is simple, lightweight and general. However, it has yet to prove that it can return results comparable to the very best of these tools.

We chose to use NICAD as our tool for evaluation, however we would expect good results using other clone detection tools [31]. File modifications and file movement was discovered using nearest-neighbour to return a single clone [32].

VII. Summary and Future Work

Our purpose was to discover how well ontology matching can be accomplished by treating it as a software-clone problem. The motivation of ontology alignment is to identify two ontologies as describing the same entity. We saw similar success in using clone detection to find WSDL clones [2], as well as in Simulink models [33]. An advantage of using clone-detection techniques lies in the fact that some of the ontology alignment algorithms can take several hours to run with small ontologies [9], whereas NICAD was designed to run on software engineering projects, which are orders of magnitude larger than the ontologies being tested. Thus is our belief that methods based on clone-detection techniques will easily scale to very large ontologies.

We plan to continue to tune our method, in particular its normalizations, and to generalize it to OWL properties and instances. Our tool does not yet perform any renaming - a feature we will implement soon - so the results in this paper are still preliminary. Our tool also ignores comments. We plan to rename the labels of entities, and provide alignments for classes, properties, and instances. We will conduct experiments on whether comments may provide information. We expect our measurements to improve as we conduct more experiments.

Using best-match clone-detection has increased precision at a small cost to recall. We plan on figuring out whether there's a way to increase recall. Traditional clone-detection has a recall rate of 0.56 for threshold 0.80, and 0.68 for threshold 0.90. It would be of interest to discover whether best-match clone detection could be modified to decrease this gap for cases where recall is more important than precision.

Other ontology datasets are available from the OAEI website. We plan to run the tool on the other datasets to see how well ontology alignment works with best-matching clone detection techniques. We will compare our results with other techniques from the 2011.5 and 2012 datasets.

VIII. Acknowledgements

This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] C. Roy and J. Cordy, "NICAD: Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization," in *Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on*. IEEE, 2008, pp. 172–181.

[2] D. Martin and J. Cordy, "Analyzing web service similarity using contextual clones," in *Proceedings of the 5th International Workshop on Software Clones, IWSC 2011*, 2011, pp. 41–46.

[3] J. Cordy, "The TXL source transformation language," *Science of Computer Programming*, vol. 61, no. 3, pp. 190–210, 2006.

[4] B. Motik, P. F. Patel-Schneider, B. Parsia, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, and M. Smith, "OWL 2 web ontology language structural specification and functional-style syntax (second edition)."

[5] J. Euzenat. Ontology alignment evaluation initiative. [Online]. Available: http://oaei.ontologymatching.org

[6] Ontology alignment evaluation initiative - oaei-2011 campaign. [Online]. Available: http://oaei.ontologymatching.org/2011/benchmarks/

[7] P. Shvaiko and J. Euzenat, "Ontology matching: state of the art and future challenges," 2012.

[8] J. Gracia, J. Bernad, and E. Mena, "Ontology matching with CIDER: evaluation report for OAEI 2011," 2011.

[9] J. Euzenat, A. Ferrara, W. R. van Hage, L. Hollink, C. Meilicke, A. Nikolov, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Šváb Zamazal, and C. Trojahn, "Fianl results of the ontology alignment evaluation initiative 2011," Tech. Rep., 2011. [Online]. Available: http://oaei.ontologymatching.org/2011/benchmarks/

[10] P. Lambrix and H. Tan, "Sambo a system for aligning and merging biomedical ontologies," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 4, no. 3, pp. 196–206, 2006.

[11] M. Nagy, M. Vargas-Vera, and E. Motta, "DSSim-managing uncertainty on the semantic web," 2007.

[12] J. Li, J. Tang, Y. Li, and Q. Luo, "RiMOM: A dynamic multistrategy ontology alignment framework," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 8, pp. 1218–1232, 2009.

[13] Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka, "Ontology matching with semantic verification," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 3, pp. 235–251, 2009.

[14] I. F. Cruz, F. P. Antonelli, and C. Stroe, "AgreementMaker: efficient matching for large real-world schemas and ontologies," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1586–1589, 2009.

[15] D. Ngo and Z. Bellahsene, "YAM++ : A multi-strategy based approach for ontology matching task," in *Knowledge Engineering and Knowledge Management*, ser. LNCS. Springer, 2012, vol. 7603, pp. 421–425.

[16] M. Kachroudi, E. B. Moussa, S. Zghal, and S. Ben, "LDOA results for OAEI 2011," *Ontology Matching*, p. 148, 2011.

[17] S. Hertling and H. Paulheim, "WikiMatch results for OAEI 2012," *Ontology Matching*, p. 220, 2012.

[18] G. A. Miller *et al.*, "WordNet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[19] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A nucleus for a web of open data," *The Semantic Web*, pp. 722–735, 2007.

[20] M. Cheatham, "MapSSS results for OAEI 2011," in *Proceedings of the ISWC 2011 workshop on ontology matching*, 2011, pp. 184–190.

[21] U. Thayasivam and P. Doshi, "Optima results for OAEI 2011," in *Proc. of 6th OM Workshop*, 2011, pp. 204–211.

[22] P. Wang, "Lily results on SEALS platform for oaei 2011," in *Proc. of 6th OM Workshop*, 2011, pp. 156–162.

[23] E. Jiménez-Ruiz and B. Cuenca Grau, "Logmap: Logic-based and scalable ontology matching," *The Semantic Web–ISWC 2011*, pp. 273–288, 2011.

[24] J. David, F. Guillet, and H. Briand, "Matching directories and OWL ontologies with AROMA," in *CIKM*, vol. 6, no. 11, 2006, pp. 830–831.

[25] J. Huber, T. Sztyler, J. Noessner, and C. Meilicke, "CODI: Combinatorial optimization for data integration–results for OAEI 2011," *Ontology Matching*, p. 134, 2011.

[26] F. C. Schadd and N. Roos, "Maasmatch results for OAEI 2011," *Ontology Matching*, p. 171, 2011.

[27] W. Hu, Y. Qu, and G. Cheng, "Matching large ontologies: A divide-and-conquer approach," *Data & Knowledge Engineering*, vol. 67, no. 1, pp. 140–160, 2008.

[28] Q.-V. Tran, R. Ichise, and B.-Q. Ho, "Cluster-based similarity aggregation for ontology matching," in *Proc. of 6th Ontology Matching Workshop*, 2011, pp. 142–147.

[29] S. Hertling, "Hertuda results for OAEI 2012," *Ontology Matching*, p. 141, 2012.

[30] J. Bock, C. Dänschel, and M. Stumpp, "MapPSO and MapEVO results for oaei 2011," *Ontology Matching*, p. 179, 2011.

[31] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of Computer Programming*, vol. 74, no. 7, pp. 470–495, 2009.

[32] T. Lavoie, F. Khomh, E. Merlo, and Y. Zou, "Inferring repository file structure modifications using nearest-neighbor clone detection," in *WCRE*, 2012, pp. 325–334.

[33] M. Alalfi, J. R. Cordy, T. Dean, M. Stephan, and A. Stevenson, "Models are code too: Near-miss clone detection for simulink models," in *Proc. of ICSM*, vol. 12, 2012.

[34] J. Euzenat and P. Shvaiko, *Ontology matching*. Springer Berlin, 2007, vol. 18.

[35] OWL2 web ontology language primer (second edition). [Online]. Available: www.w3c.org/TR/2012/REC-owl2-primer-20121211/

[36] S. Bechhofer, *et al.*, "OWL web ontology language reference," *W3C recommendation*, vol. 10, pp. 2006–01, 2004.

[37] D. L. McGuinness *et al.*, "OWL web ontology language overview," *W3C recommendation*, vol. 10, no. 2004-03, p. 10, 2004.