CROSI Mapping System (CMS)
Results of the 2005 Ontology Alignment Contest

Yannis Kalfoglou
Advanced Knowledge Technologies (AKT)
School of Electronics and Computer Science
University of Southampton, UK
y.kalfoglou@ecs.soton.ac.uk

Bo Hu
School of Electronics and Computer Science
University of Southampton, UK
bh@ecs.soton.ac.uk

ABSTRACT
In this results report we summarize our experiences from running the CROSI Mapping System (CMS) over three test cases for this year's OAEI contest: bibliography, Web directories and medical ontologies alignment case studies. CMS successfully parsed and aligned all input ontologies in all three case studies. We also elaborate on the insights gained and potential research directions towards building more robust alignment systems to cope with the increasing diversity of alignment requirements.

1. PRESENTATION OF THE SYSTEM
The CROSI Mapping System (hereafter, CMS) has been developed in the context of the CROSI project (which stands for Capturing Representing and Operationalising Semantic Iinteroperability). CROSI, which is funded by HP, started in November of 2004 and will run until November of 2005[1]. It aims to develop a systematic approach upon which semantic interoperability can be studied and operationalised by (a) capturing and exposing semantics, (b) codify them in Knowledge Representation formats, and (c) operationalise them for the benefit of integration. One of the CROSI deliverables that we used in the early stages of the CMS design, was the notion of *semantic intensity spectrum*[2] which helped us identify what kind of tools and algorithms we could employ in CMS for the OAEI contest. These were used in a modular architecture we devised, reminiscent of similar architectures proposed in the past (see, for example, [4]), which we depict schematically in figure 1

[1]more can be found at: `www.aktors.org\crosi`
[2]more on: `www.aktors.org\crosi\si-spectrum`

In the core of this architecture lies the CMS system. CMS is a structure matching system that capitalizes on the rich semantics of the OWL constructs found in source ontologies and on its modular architecture that allows the system to consult external linguistic resources.

Most of these resources use various families of algorithms which aim to compute similarity based on string distance, e.g. SecondString packages [1]. String distance is one of the widely used techniques in finding correspondences between ontologies. It normally takes as input the names of two concepts calculating the distance, by editing the distance in its simplest form or hybrid distance functions in a more sophisticated form, and output a numeric value to represent the confidence of the similarity. Sometimes, natural language processing methods are employed to cut down the number of string tokens that need to compute the similarity for.

However, string similarity is not sufficient to capture the subtle differences between classes with similar names but different meanings and it can produce misleading results. To alleviate the situation we can work with Natural Language Processing (NLP) packages that exploit synonymy at the: 1) lexical-level, e.g. the use of WordNet [3] to provide a source of synonyms, hypernyms and hyponyms; the 2) phrase- and sentence-level, e.g. phrases and sentences in the active and passive forms but having the same meanings; and the 3) semantic-level synonymy.

Although WordNet-based approaches equip themselves with the lexical synonymy of the names of classes, they do not have the right measure to capture the structural information that is conveyed in most taxonomies. Structural information is exploited in different ways. Heuristic rules is the most common way to take structures into account, e.g. identifying similarity of two entities based on the status of their parents and siblings.

The modular architecture depicted in figure 1 employs a multi-strategy system comprising of four modules, namely, *Feature Generation*, *Feature Selection and Pro-*

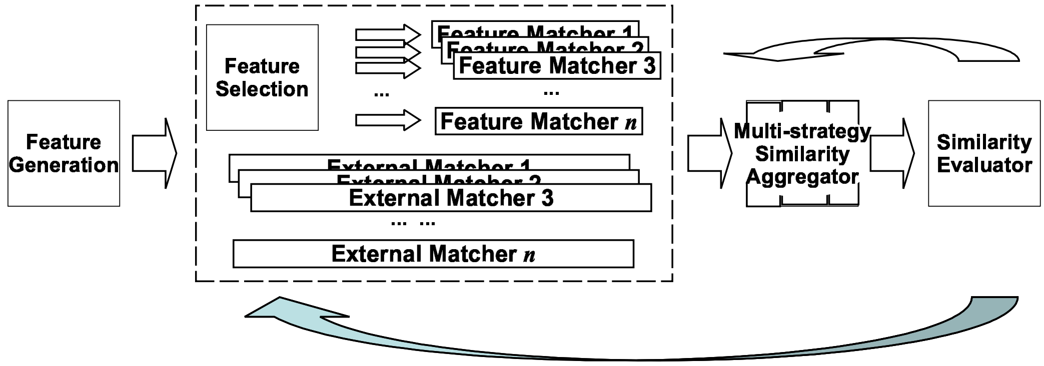

Figure 1: CMS ontology alignment system and its modular architecture.

cessing, *Aggregator* and *Evaluator*. In this system, different features of the input data are generated and selected to fire off different sorts of feature matchers. The resultant similarity values are compiled by multiple similarity aggregators running in parallel or consecutive order. The overall similarity is then evaluated to initiate iterations that backtrack to different stages.

CMS, is an instantiation of such a system. We include a screenshot of the Web-based interface of CMS in figure 2. The system is still under development and we only used the first two components, *Feature generation* and *Feature Selection and Processing*, for aligning the ontologies in the three case studies of the OAEI contest. The alignment algorithms and techniques used are described in later sections but first we elaborate, in the next section, on the purpose of CMS and highlight some of its key characteristics, like the robust features extraction module.

1.1 State, purpose, general statement

The process of ontology mapping (or alignment), can be summarised as: given two ontologies, a system measures the similarity of the source ontological entities against the target ones and produces a list of correspondences, i.e. $\texttt{mapping} : O_s, O_t \rightarrow \mathcal{C}_s \times \mathcal{C}_t \cup \mathcal{P}_s \times \mathcal{P}_t \cup \mathcal{I}_s \times \mathcal{I}_t$ where O_i is the input ontologies with $i \in \{s,\ t\}$, subscript s indicating the source and t indicating the target, $\mathcal{C}_i$ the set of classes, $\mathcal{P}_i$ the set of properties and $\mathcal{I}_i$ the set of instances. Hence, the first step when deploying CMS was to extract characteristics that can be used to identify similar entities from different ontologies. We summarize the characteristics we extracted in table 1.

There are several points that need further explanation. First, in many cases, identifying corresponding instances is considered to be an easier task than identifying corresponding classes. This is because instances are expected to have more grounded variables. Corresponding instances provide a ground on which the number of candidate mapping classes can be narrowed down to a few (as we discovered in our past work with the IF-Map instance-based system [?]). Second, in case of complement classes, let c_s be a class from the source ontology and c_t from the target ontology, if $sim(c_s, c_t) = a$ and $d = \neg c$, we can safely conclude that $sim(d, c_s) = 1 - a$, where $sim/2$ is the similarity function and a, a real number, gives the confident value.

1.2 Specific techniques used

To fit the requirements of different applications, we developed and implemented a series of mapping techniques, which are regarded as independent components that made up the CMS.

Name matchers

Ranging from pure syntactical approaches to more semantic enriched ones, name matchers are categorised as: String (tokenised) distance, Thesaurus, and WordNet hierarchical distance.

Levenstain distance is the simplest implementation of string distance. More sophisticated ones are: *Monge-Elkan* distance optimises edit-distance functions with well-tuned editing cost and *Jaro* Metric and its variants computes an accumulated similarity of s and t from the order and number of common characters between s and t, just to name a few. In our system thesaurus comes into play in two forms: WordNet[3] and a predefined corpora that are implemented as `WNNameMatcher` and `CorpusNameMatcher` respectively. To facilitate the use of WordNet, we assume that the local names of classes are either nouns or noun phrases while the local names of properties are phrases starting with verbs followed by either nouns or adjectives. Elements in the retrieved synsets are then compared against each other using either exact string matching or one of the string-distance based algorithms discussed in the previous section. WordNet arranges it entries in hierarchical structures. Hence, the similarity between names can be computed as followings: let w_i and w_j be the corresponding WordNet entries of $name_i$ and $name_j$, w

[3] http://wordnet.princeton.edu/

Local features

class *labels* and *URIs*	classes with same local names but different name spaces need to be treated with caution, as there is a risk that they might be different in different contexts.
equivalent classes	equivalent classes give the alternatives of a class that can be regarded as hints for identifying new mapping candidates.
related *property names*	both declared and inherited properties contribute to the meaning of a class and thus should be extracted.
complement classes	complement classes indicates semantic dissimilarity.
property *labels* and *URIs*	same as for classes.
property *domain* and *range*	the domain and range of a property can pin down the meaning of a class when name matching is not sufficient.
inverse (transitive) property	both inverse and transitive properties are regarded as hints for similar properties and thus indirect hints for similar classes.
functional property	functional properties play the same role in identifying corresponding classes as *keys* do in element level database schema matching.
instance *labels* and *URIs*	same as for classes.
instantiated classes	instances are treated as a source of understanding semantics.
comments	well documented design rationale is a reliable source for revealing semantics.

Global features

super and *sub classes*	subsumption relationship help to identify the location of a class in the taxonomy and thus capture the structural semantics.
sibling classes	sibling classes provide the hint of how the parent class is defined.
super and *sub properties*	properties' hierarchy is useful in matching both properties and classes
disjoint classes	disjoint cover should be treated as a special case.
comments	comments sometimes are also given at the global level.
version information	the record of modifications and authentication provides alternatives.

Table 1: Features extracted for Ontology Mapping.

be the least common hypernym of w_i and w_j, r be the root of the underlying WordNet hierarchy, and h_i, h_j, h be the distances between w_i and r, w_j and r, w and r, respectively, the similarity between w_i and w_j is approximated as $2 \times h/h_i + h_j$.

Semantic matchers

In CMS, the flavour of semantic is added in two different ways: namely structure-aware matchers and intension-aware matchers.

Structure-awareness refers to the capability of traversing class hierarchies and accumulating similarities along the sub-class (sub-property) relationships. Let c and d be two classes from source and target ontologies, c_i and d_i are their direct parents in respective ontologies, the similarity between c and d is recursively defined as $\mathtt{sim}(c,d) = \alpha\mathtt{sim}_{local}(c,d) + \beta\mathtt{sim}(c_i,d_i)$, where α and β are arbitrary weights and $\mathtt{sim}_{\mathrm{local}}/2$ gives the local similarity with regard to c and d which can be computed using one or a combination of techniques discussed above.

Intension-awareness takes into account the definitions of classes. A class c are regarded as a tuple $\langle S, P\rangle$ where S is a set of classes of which c is a subclass and P is a set of properties having c as the domain and other classes or concrete data types as the range. Hence, finding the semantic similarity between $c = \langle S_c, P_c\rangle$ and $d = \langle S_d, P_d\rangle$ amounts to finding the similarity between S_c and S_d as well as P_c and P_d, i.e. $\mathtt{sim}(c,d) = \alpha\mathtt{sim}(S_c,S_d) + \beta\mathtt{sim}_{property}(P_c,P_d)$, where α and β are arbitrary weights and $\mathtt{sim}_{property}/2$ computes the property similarity. More specifically, we differentiate the following situations:

- classes with matching property names, property domains and property ranges: $\mathcal{L}_{p_c} = \mathcal{L}_{p_d}$ and $sim_{set}(\Delta_{p_c}, \Delta_{p_d}) \geq v$ and $sim_{set}(\Phi_{p_c}, \Phi_{p_d}) \geq v$ where $sim_{set}/2$ computes the similarity of two sets of entities and v is a predefined threshold.
- classes with matching property names and property domains but different property ranges: $\mathcal{L}_{p_c} = \mathcal{L}_{p_d}$ and $sim_{set}(\Delta_{p_d}, \Delta_{p_d}) \geq v$, $sim_{set}(\Phi_{p_c}, \Phi_{p_d}) < v$, and
- classes with matching property names but different property domains as well as ranges: $\mathcal{L}_{p_c} = \mathcal{L}_{p_d}$ and $sim_{set}(\Delta_{p_c}, \Delta_{p_d}) < v$ and $sim_{set}(\Phi_{p_c}, \Phi_{p_d}) < v$.

The first situation contributes the most to the similarity of c and d. We regard classes with matching names and exact matching properties, i.e., properties with same name, domain and range, as semantically equivalent classes.

In many cases, matching between Δ_{P_c} and Δ_{P_d} (Φ_{P_c} and Φ_{P_c}, respectively) can only be concluded after traversing several levels upwards or downwards the class hierarchy. Although not as strong as exact matching of property domains and ranges, matching classes of Δ_{P_c} (Φ_{P_c}) to remote ancestors or descendants of classes of Δ_{P_d} (Φ_{P_d}) provides a hint on how close the different properties are, and thus how similar the two concepts c and d are. Such an idea is implemented in our system as a `ClassDefPlusMatcher` method.

1.3 Adaptations made for the contest

We didn't do any major adaptations to CMS in order to align the OAEI contest ontologies. We only did minor, routine programmatic adjustments, as for example running the CMS system from the command line prompt in a batch mode to parse and align the hundreds of ontologies in the Web directories case or include specific Java heap size adjustment flags in order to run the system over the vast FMA ontology. Other than that, the system ran as normal.

2. RESULTS

CMS benefits from the plug and play of modular matchers. In this contest, four different matchers were used, namely `ClassDef` for examining the domain and range of properties associated with classes, `CanoName` for accumulating similarities among class hierarchies, `WNDisSim` for computing the distance between two class names based on WordNet structures and `HierarchyDisSim` for distributing similarity among class hierarchies. The four major matchers were invoked both in parallel and sequentially. When invoked in parallel their results were then aggregated as weight average. On the other hand, when invoked in sequence, `CanoName` and `WNDisSim` give a list of corresponding classes whose similarities were then refined by `ClassDef` and `HierarchyDisSim`. CMS ran each test case with different configurations (combination and sequencing) of the aforementioned four mapping modules and precision and recall values were calculated for each run. In this report, we include the the configurations with the highest precision and recall values.

2.1 Case 1: benchmark/BibTex ontologies

For all the ontologies in this case we used a threshold of 0.8.

ontology 202: CMS fails to produce any mapping candidates with high similarity score in test case 202 due to the naming convention. We consider class names as the foundation on which other techniques can be applied (although not the sole and dominant clue for finding mapping candidates). Similarly, cases 248 to 266 also fall into this category: no candidates with high similarity value were found.

ontology 205: CMS does not achieve a high recall rate for benchmark test case 205 due to the restriction of WordNet. In case 205, class names are replaced by randomly selected synonyms. CMS relies heavily on external resources, e.g. WordNet, to provide lexical alternatives for class and property names and thus fails to respond well for synonyms that are not recognised by WordNet. A customised corpus might alleviate the problem and improve the performance with significant efforts and domain expertise.

ontology 301: In test case 301, smaller similarity scores were assigned to mapping candidates. This is due to the fact that although classes have similar names, they are defined with different properties which have different names, domains and/or ranges. It is our contention that for classes restricted with different properties, they should either not be considered as equivalent classes or their similarity value should be reduced to reflect such difference.

2.2 Case 2: Web directories ontologies

We do not have any specific comments for Case 2. All 2265 were parsed successfully by CMS and fetched for alignment. However, 29 ontologies did not produced any alignment results due to circular definitions in the original `source.owl` and `target.owl` files. So, a total of 2236 pairs of `source.owl/target.owl` were aligned. The system parsed them from the command line in a batch mode, and the results produced after 2 hours and 53 minutes. Each cycle involved reading and parsing the source and target ontologies, find alignments (if any) and save and write the results in the common alignment format in a file. This was repeated 2265 times.

2.3 Case 3: Medical ontologies

This case was the most interesting. The sheer size of the input ontologies (especially that of FMA), the modelling style of OWL, the conventions used, and the complexity of the paradigm made it an interesting adventure from the research point of view. We report in more detail about our experiences in section 3.3.

3. GENERAL COMMENTS

Performance tuning and hardware settings: As we were facing some really large ontologies (i.e., the 72k classes FMA ontology), we had to do certain optimizations to the code and to the computer settings in order to obtain alignment results in acceptable time. We ran the tests on a stand-alone PC running Microsoft Windows XP operating system, service pack II, 2003 version. The PC had 1GB of memory installed (DDR400-SDRAM), an 80GB Serial ATA hard disk, and a Pen-

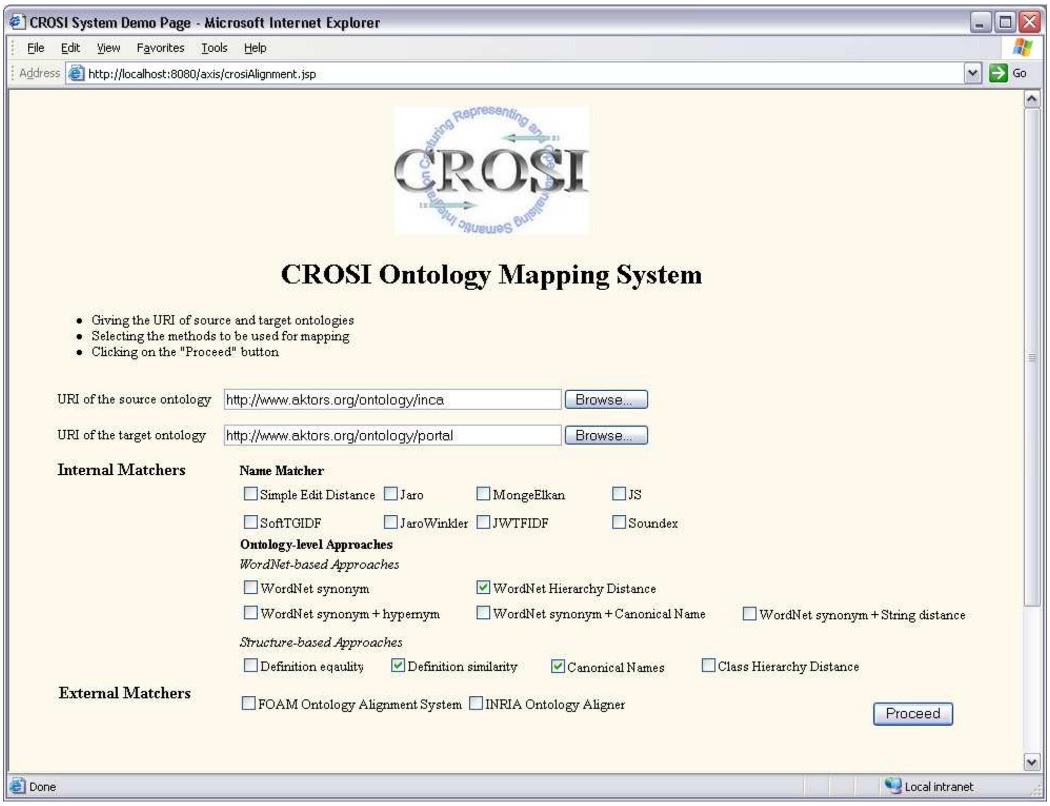

Figure 2: Web-based Interface for CMS.

tium 4, 3.0GHz processor. We used Java VM (version 1.5.0_04) and we had to do certain configurations to adjust the heap size in Java. For example, the standard Java heap size is 64MB. This was not enough though for the Web directory and medical ontologies case. In fact, for the medical ontologies case, the sheer size of the input ontologies (especially that of FMA) forced us to use a 768MB heap size. Settings lower than this threshold caused the system to run out of memory.

Parsing and extracting experiences: FMA owl is a 31MB .owl file comprising of 72545 declarations of owl classes and 100 relations (object and data type properties). These numbers were obtained when using our Jena 2.2 API and probably deviate slightly from other parsers. Parsing and extracting features from the FMA ontology took 9 minutes and 17 seconds with Java Heap Size adjusted to 512MB. However, in order to run the CMS and find alignments with the OpenGALEN we had to use a 768MB heap size setting. While parsing, Jena API was complaining about the syntax idioms used. For example we had a lot of warnings from Jena's RDF syntax handler, or the form "bad URI in qname XXX: no scheme found". We elaborate on the reasons behind this parsing warnings in section 3.3.

OpenGALEN.owl is a 4MB .owl file comprising of 24 declarations of owl classes and 30 relations (as previously, object and data type properties, and these numbers were obtained from Jena 2.2 API). Parsing and extracting features from OpenGALEN took just a few seconds. There was no need to adjust the Java heap size.

3.1 Comments on the results

Different combinations of CMS plug-in matchers perform significantly differently due to the nature of benchmark test cases. Table 3.1 lists the choice of matchers with regard to each test cases while Table 3.2 shows performance values of different matchers[4] with regard to alignment of ontology 303 in case 1, in terms of precision and recall.

3.2 Discussions on the way to improve the proposed system

CMS is expected to be improved on the following aspects: a more sophisticated aggregation mechanism, a unified alignment representation formalism, and parameterised algorithms for class hierarchy distance.

Firstly, as discussed in previous sections, results from multi-matchers are aggregated as weighted average with arbitrary weights to start with. Thus far, the weights are fine-tuned manually relying on the knowledge of the

[4]Results are obtained with equal weights for matchers.

CMS Matchers	Test Case #
A	103, 201, 210,
A, B	205, 206, 207, 209, 301, 303
A, C, D	225, 228, 233, 236, 239-241, 246, 247, 248-266, 302
A, B, C, D	104, 203, 204, 208, 221, 222, 223, 224, 230, 231, 232, 237, 238, 304

A–Class Definition,
B–Canonical Name,
C–WordNet Hierarchy Distance,
D–Class Hierarchy Distance

Table 2: CMS matchers combinations.

CMS Matchers for #303	Precision	Recall
Class Definition (A)	0.6923	0.4736
Canonical Name (B)	0.3243	0.6315
WordNet (WN) synonym	0.06	0.7894
WN Hierarchy Dis (C)	0.24	0.3157
Class Hierarchy Dis (D)	1.0	0.5263
WN synonym + hypernym (E)	0.04	0.8421
A + B	0.9	0.4736
A + E	1.0	0.4736
A + B + E	1.0	0.4736
A + B + D	1.0	0.3684
B + C + D	0.8	0.4210
B + C + D	0.8	0.4210

Table 3: Performance of different matchers for test case #303.

domain of discourse and the underlying algorithms of CMS. A more sophisticated approach would hire machine learning techniques to work out the most appropriate weights with regard to different matchers aiming to solve different sort of mappings. Furthermore, results from different matchers can be sorted locally first which could make accumulating results from different matchers to be reduced to ranking aggregation [2].

Secondly, the heterogeneous nature of different matchers – some external matchers produce pairwise equivalence with numeric values stating the similarity score while others output high level relationships, e.g. *same entity as*, *more specific than*, *more general than* and *disjoint with* expressed in high level languages such as OWL and RDF – suggests that output from different matchers has to be lifted to the same syntactical and semantic level. A unified representation formalism equipped with both numeric and abstract expressivity can facilitate the aggregation of heterogeneous matchers.

Thirdly, CMS takes into account the exact position of classes in the class hierarchy. We would like to develop algorithms that penalise mapping candidates that are found to be quite apart from each other, and then propagate their similarity values upwards and downwards in the hierarchy to their descendants and/or ancestors. There could also be pre-defined parameters that as we go up or down the hierarchy we change the similarity values of their descendants and/or ancestors accordingly. We expect that this could reduce the number of false positive results.

3.3 Comments on the test cases

We do not have any specific comments for test cases on BibTex and Web directories alignments. However, we found interesting the last test case, that of medical ontologies alignment, and we summarize our experiences below.

FMA.owl was a different case altogether. The ontology describes the domain of human anatomy and it aims to provide "a reference ontology in biomedical informatics for correlating different views of anatomy, aligning existing and emerging ontologies in bioinformatics" [6]. However, there are two notable facts regarding the syntactic and modelling idioms of FMA and existing results from previous efforts in trying to align FMA and GALEN. As far as the former is concerned, the OWL version we had to work with was a result of translation from Protege. Previous work has shown that this result is not always a faithful representation of the original FMA Protege model. For instance, it has been reported that FMA DL constructs are often ill-defined and they lead to inconsistencies when a reasoner parses the ontology [5]. Consistency checking for FMA is an acknowledged problem though, even by its authors: "[...] feedback from these investigators revealed an aggregate of a few hundred errors, many of which related to spelling and only a few to cycles in the class subsumption and partonomy hierarchies." [6].

Leaving aside this fact of life (as it is natural for an ontology that big and so close to human practice to be inconsistent), we point to a couple of syntactic idioms that we found interesting when parsing the ontology with our Jena-based CMS system. Firstly, the rather unusual use of unique frame IDs for class names (`<owl:Class rdf:ID>` constructs) and the textual description of a class in an `rdfs:label` construct. We also noticed some unusual uses of references to frame IDs. For instance, the declaration of "arterial supply" as an object property: `<owl:ObjectProperty rdf:ID="arterial_supply" rdfs:label="arterial supply">` is used in other parts of the ontology where it refers to a `rdf:resource` which points to a different resource:
`<arterial_supply rdf:resource="#frame_14586"/>`.
Tracing that frame ID leads us to a definition of a "Tissue" class, and not the "arterial supply": `<owl:Class rdf:ID="frame_14586" rdfs:label="Tissue">`. The definition of an instance (with frame ID 14586) of an ob-

ject property ("arterial supply") that is a class ("Tissue") could lead to modelling misunderstandings and confusion (although, syntactically speaking, it is allowed in some versions of OWL).

Going back to our argument for the notable facts, we found that previous efforts for aligning FMA to GALEN reported rather controversial results. For example, in [7], the authors employed two different alignment methods to map FMA to GALEN. Despite of the subtle differences of OpenGALEN with GALEN, the similarity of their work with that of the OAEI contest 3rd case study is high but some of their findings are questionable from the semantics point of view: for example, it was reported that "Pancreas" in FMA matches "Pancreas" in OpenGALEN with 1.0 similarity value which "indicates a perfect match" [7]. When we looked carefully at the definitions of "Pancreas" in both ontologies we saw that "Pancreas" is defined as a class in FMA (<owl:Class rdf:ID="frame_12280" rdfs:label="Pancreas">) whereas in GALEN (OpenGALEN) as an instance of class "Body Cavity Anatomy"

```
<owl:Class rdf:ID="Body_Cavity_Anatomy">
<rdfs:subClassOf
rdf:resource="#OpenGALEN_Anatomy_Metaclass"/>
<Body_Cavity_Anatomy rdf:ID="Pancreas">
```

Even if OWL semantics allow to map an individual to a class (when dealing with OWL Full), such an alignment is misleading especially when we consider the high level of abstraction for the "Pancreas" class in OpenGALEN. It seems that the "lexical phase" parsing used in [7] was the main contributor to this high similarity value when relatively little structure information was taken into account. As a final comment on the case, we also point the reader to observations made by the FMA authors when trying to validate mapping results and differences in terminologies with these two ontologies: "[...]the reasons for the differences have not yet been explored, but at least some of them may be the different contexts of modelling. GALEN represents anatomy in the context of surgical procedures, whereas FMA has a strictly structural orientation." [6].

3.4 Comments on the measures

The proposed measures of precision and recall have been studied and practiced in the NLP community for years and they are a *de facto* standard metric for commercial applications, like search engines. However, we believe that their adaptation for measuring the performance of an ontology mapping system is somewhat questionable. We cannot elaborate fully on our reservations regarding the use of such a metric in this short paper, but we highlight the main points of our objections: (a) precision is regarded as hard to implement and reveals the usefulness of a retrieved document (or hit in a hitlist) for a search engine. We can't judge the usefulness of a found alignment by comparing it with the reference alignment; (b) neither precision nor recall take into account the possible applications of the alignments found. In all the past EON (and this year OAEI) contests, a set of pre-defined alignments were used as a standard against which all found alignment were compared. This does not say anything about the usefulness of the found alignments, or even of they are complete as the pre-defined ones can be erroneous. Further to these comments, we would also like to add that the assignment of numerical values in the range 0.0 to 1.0 does not reveal their semantic relevance, but purely a brute-force algorithmic way of comparing performance. We also observed a variety of interpretations of precision and recall metrics by the ontology alignment community.

3.5 Proposed new measures

Devising new measures for assessing the found alignments between two ontologies in a universally agreed manner is a difficult task. We do not see a quick solution to this problem, but as ontology engineers we can apply knowledge engineering technologies that encompass as much semantic information as possible; for example, we were surprised that the semantically rich definitions of OWL for declaring class or property equality (and inequality) and the universal construct for declaring similarity, are hardly used by the community.

We would also like to see ways of introducing "application-driven" alignment metrics where an example application (i.e., a Semantic Web service information lookup engine) will need to access two different ontologies and the alignments found will need to be used in the application in a specific way. Having an application-driven alignment metric, we can experiment with the notion of usefulness of alignment in a real world scenario, rather than doing meaningless number crunching with regard to found and pre-defined alignments. After all, alignment needs to be done in the first place because there is a real world need for it.

4. CONCLUSION

The 2005 OAEI ontology alignment contest was the first one that introduced sizeable ontologies and posed some interesting and challenging problems with respect to performance, scaling and domain exploration. We found it a rewarding experience and we look forward to continue the fruitful exploration of this key field in the emergent Semantic Web.

5. REFERENCES

[1] W.W. Cohen, P. Ravikumar, and S.E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IJCAI 2003 IIWeb Workshop*, pages 73–78, 2003.

[2] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the ACM SIGMOD*

International Conference on Management of Data, pages 301–312. ACM Press, 2003.

[3] C. Fellbaum. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

[4] M. Ehrig and S. Staab. QOM - Quick Ontology Mapping. In *Proceedings of the 3rd International Semantic Web Confernece (ISWC'04), LNCS 3298, Hiroshima, Japan*, page 683–697, 2004.

[5] C. Golbreich, S. Zhang, and O. Bodenreider. Migrating the FMA from Protege to OWL. Technical report, jul 2005. In notes of the 8th International Protege Conference, Madrid, Spain.

[6] C. Rosse and JL. Mejino. A Reference Ontology for Bioinformatics: The Foundational Model of Anatomy. *Journal of Biomedical Informatics*, 36:478–500, 2003.

[7] S. Zhang, P. Mork, and O. Bodenreider. Lessons learned from aligning two representations of anatomy. In *in Proceedings of the KR 2004 Workshop on Formal Biomedical Knowledge Representation, Whistler, BC, Canada*, pages 102–108, 2004.

6. RAW RESULTS

All of our results are included in a tabular format in table 6.3. These results have been the best of the CMS combinations with different matcher. We report on those in section 3.1. So, for example, alignments for case #103 were produced using CMS Matcher A, whereas alignments for case 225 were produced using CMS Matchers A+B+C. A list of all this combibnation can be found in table 3.2.

6.1 Link to the system and parameters file

Access to the Web-based interface of the CMS system is provided via `www.aktors.org/crosi/cms`. We note that the system is not available in the community for free distribution yet, due to the legalities of the IPR for the CROSI project.

6.2 Link to the set of provided alignments (in align format)

The results of all three cases (BibTex, Web directories, Medical) are available for download from the CROSI web site at `www.aktors.org/crosi/eon05contest/results`.

6.3 Matrix of results

#	Name	Prec.	Rec.	Time (s)
101	Reference alignment	N/A	N/A	N/A
102	Irrelevant ontology	N/A	N/A	108
103	Language generalization	1.0	0.788	88
104	Language restriction	1.0	0.788	159
201	No names	1.0	0.189	70
202	No names, no comments	N/A	N/A	
203	No comments	1.0	0.697	147
204	Naming conventions	1.0	0.605	153
205	Synonyms	1.0	0.230	85
206	Translation	1.0	0.255	82
207		1.0	0.264	88
208		1.0	0.473	149
209		1.0	0.103	84
210		0.818	0.246	74
221	No specialisation	1.0	0.788	129
222	Flatenned hierarchy	1.0	0.724	169
223	Expanded hierarchy	0.962	0.758	316
224	No instance	1.0	0.788	151
225	No restrictions	0.788	0.788	85
228	No properties	0.788	0.788	76
230	Flattened classes	1.0	0.760	161
231	Expanded classes	1.0	0.788	145
232		1.0	0.788	118
233		0.838	0.788	70
236		0.788	0.788	77
237		1.0	0.724	156
238		0.961	0.757	315
239		0.766	0.793	220
240		0.757	0.757	221
241		0.838	0.788	70
246		0.766	0.793	70
247		0.757	0.757	221
301	Real: BibTeX/MIT	1.0	0.363	30
302	Real: BibTeX/UMBC	1.0	0.348	31
303	Real: Karlsruhe	1.0	0.474	328
304	Real: INRIA	0.85	0.566	131

Acknowledgements

This work is supported under the Capturing, Representing, and Operationalising Semantic Integration (CROSI) project which is sponsored by Hewlett Packard Laboratories at Bristol, UK. The first author is also supported by the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC) project which is sponsored by the UK EPSRC under Grant number GR/N15764/01.