Targeted Ontology Mapping

Michelle Cheatham
The Design Knowledge Company
michelle.cheatham@gmail.com

ABSTRACT

We examine the ontology mapping problem with three different groups of readers in mind: system designers, who need to know what is and isn't possible given the current state of alignment algorithms and what pitfalls to watch out for when someone suggests the use of ontologies in a project; system developers, who require knowledge of specific algorithms and useful software libraries; and semantic interoperability researchers who may be interested in the potential efficacy of using a genetic algorithm to choose which similarity metrics should be employed to map two ontologies, based on a small, carefully-chosen subset of each ontology.

KEYWORDS: ontology alignment, ontology mapping, genetic algorithm

1. INTRODUCTION

In this work we focus on the ontology mapping (or alignment) problem. When two individuals are asked to develop an ontology to describe an area of interest (food, for example), it is very unlikely that they will produce identical results. As a simple example, one may have a class called "fruits and vegetables" while the other has a class called "produce", but both of these serve the same role in their respective ontologies. In order for ontologies to be useful in applications requiring semantic integration, we need a method of determining that when agent A says "fruits and vegetables", agent B should hear "produce." Without this functionality it is not possible for the two agents (which could be humans or software) to collaborate effectively.

The essence of the ontology mapping problem is to take two ontologies and determine which entities in the first correspond in meaning to which entities in the second. This can be formally defined as follows: let us assume we have two ontologies O1 = {C_1, R_1, I_1, A_1} and O2 = {C_2, R_2, I_2, A_2}; each ontology is a set of classes, relations, instances, and axioms. We wish to define a mapping function m(e_1, e_2, c) where e_1 is any element from O_1, e_2 is an element from O_2, and c is the confidence we have that this particular mapping is correct. The general approach is to choose mappings between e_1 and e_2 such that one or more metrics indicate that the two entities are more similar to one another than they are to any other entities and the similarity value is above some validity threshold. In this work entities will be mapped one-to-one. Figure 1 shows a depiction of two ontologies and some potential mappings between them.

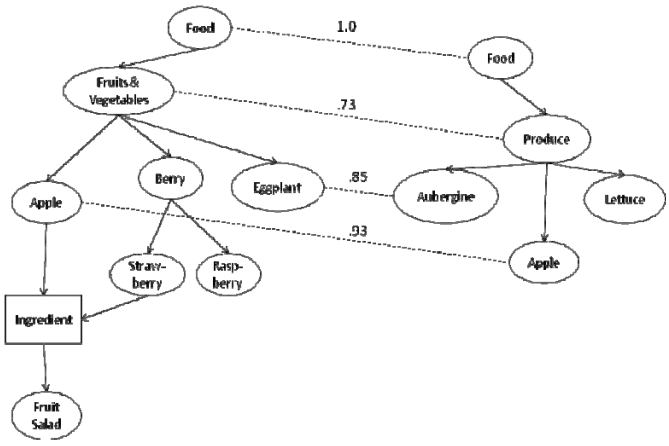

Figure 1. Ontology Mapping

A variety of issues combine to make the ontology matching problem a tough nut to crack. As mentioned previously, users may choose different labels to represent the same concept, a situation referred to as synonymy. Conversely, polysemy refers to instances in which the same word refers to two different concepts, as in the case of "date," which can be a piece of fruit, a specific moment in time, or a rendezvous. Different users may choose different units of measure for data values. They may consider different details to be irrelevant or think at different levels of abstraction. They may reify (consider

as an entity) different concepts. For instance, one person may consider red and yellow apples to be different types of the same entity, while another may consider them distinct entities that are descended from the same parent. Additionally, the ontologies may be large enough that some mapping approaches are not computationally feasible. Pavel Shvaiko and Jerome Euzenat provide much more detail on this subject in [4], in which they describe what they consider to be the top ten challenges for ontology matching. Our work described here focuses on the problem mentioned in Section 9 of that paper, matcher selection and self-configuration. This challenge concerns choosing the best matching algorithms, in the best combination, with the best parameters for the particular task at hand.

As we will see in a later section, many existing ontology mapping algorithms perform well in the face of some of the confounding issues mentioned above but none has been shown to satisfactorily generate mappings in a wide enough set of situations to be practically useful in many applications. Their performance varies, often substantially, based on the particular pair of ontologies under consideration. In this research effort we consider whether it is possible to reduce this variation by carefully selecting a small subset of both ontologies and using a genetic algorithm to determine the best set of mapping approaches to apply in this particular case. If successful, this technique would be another step towards making ontologies useful for collaboration in practical applications.

Ontology alignment has many applications within human-machine and machine-machine collaborative systems. It can be used to merge information from various data sources, facilitate communication between software agents, and aid in web service composition, among many other possibilities.

2. CURRENT APPOACHES

Mark Ehrig and Steffen Staab, in [5], define a useful overview of the ontology mapping process that we will use to review current approaches. Their view of the generic iterative process consists of five steps, to which we have added alignment debugging, a relatively recent advancement in the field.

1. Feature Engineering: input the ontologies and extract the features that will be used in the later steps.
2. Selection of Next Search Steps: determine the subset of elements in O_1 and O_2 that will be processed in successive steps (in the boundary case, every element in O_1 will be compared with every element of O_2 to test if they match).
3. Similarity Computation: execute one or more similarity metrics on the candidate mappings.
4. Similarity Aggregation: combine the results of multiple similarity metrics to compute a single similarity value.
5. Interpretation: decide on the mappings using the aggregation results (this may involve filtering out some mappings that have low similarity values and/or determining the best mapping when there are several potential choices with large similarity values).
6. Alignment Debugging: sanity checks identify and filter out incongruous mappings.

2.1. Feature Engineering

The first phase of the feature engineering step – reading in the ontologies – is relatively straightforward. There are several open source libraries available for this purpose, the most popular of these being a java library called JENA [12]. There are some problems that arise when dealing with extremely large ontologies that will not fit in memory or with ontologies that make frequent references to outside namespaces (so that a complete approach would be to fetch all of the referenced ontologies as well). These issues are a relatively open area of research. In contrast many researchers have performed analyses of the second phase of this step, determining which features are most useful to extract from the ontologies. These features can be classified into two groups: those intrinsic to an entity versus those that are extrinsic. Intrinsic features include the name given to an entity, its URI, its type (or superclass), the domain and range of properties, data types and values, etc. Examples of extrinsic features are subclasses, ancestors, descendents, and level/depth in the ontology, among others. Because intrinsic features are local to the entity and can be easily retrieved rather than requiring computation or extensive searches of the ontology, they are preferable to extrinsic features when they perform well.

2.2. Selection of Next Search Steps

The selection of next steps is another aspect of the difficulty in dealing with large ontologies. The default

approach mentioned above – comparing every entity in the first ontology to every entity in the second ontology – is thorough but does not scale well, particularly when considering that some of the similarity metrics may be expensive in their own right. Some algorithms attempt to reduce the number of candidate mappings considered based on some filtering criteria. Arguably, some method of reducing the number of comparisons performed *must* be employed by an alignment scheme in order for it to be practical in real-world situations. Some obvious choices for heuristic filters include considering only those entities whose labels have a reasonable level of lexical similarity, examining entities which are at similar depths in the ontological hierarchy, or having the current iteration examine entities that border mappings found during the previous iteration. Ehrig and Staab mention other possibilities in [5]. Their approach, QOM, uses different heuristics during different iterations. The first iteration uses lexical similarity of labels. Subsequent iterations expand on mappings found during previous rounds. Finally, a random strategy is pursued. Lily, an ontology alignment approach developed by Wang and Xu, also uses heuristics to reduce the search space [3].

2.3. Similarity Computation

Researchers have considered a very wide array of similarity metrics. These generally fall into three groups: lexical, semantic, and set-based. Lexical metrics compare two strings. The most common such metric is the Levenstein distance, which is given by the minimum number of single-character insertions, deletions, or substitutions necessary to transform one string into another. The Levenstein distance between "hello" and "help" is two because **p** must be substituted for **l** and the **o** must be deleted. A similar metric, developed by Jaro, considers transpositions in addition to insertions and deletions [29]. Winkler builds on this idea further by proposing a modification of the Jaro distance metric that considers strings more similar based on the length of their prefixes that are identical. The JaroWinkler metric is discussed more in [7]. Stoilos, Stamou, and Kollias argue in [6] that because the vast majority of lexical metrics were not developed with ontology mapping in mind, they are not ideally suited for that application. They propose an alternate metric that is designed to be fast, tolerant to non-optimal threshold values, and unlikely to produce the same similarity value when a string is compared to two different strings. The metric is a combination of a commonality measure (based on repeated application of a substring metric), a difference measure (based on the Hamacher triangular norm), and the JaroWinkler metric. There are a slew of other lexical distance metrics, many of which are available in the open source SecondString library [13].

Semantic similarity metrics attempt to use information about words beyond their spelling when computing similarity. The majority of semantic metrics are currently based either on the frequency and co-occurrence of words in a large corpus of documents or on a human-generated taxonomy such as WordNet. Resnik introduced a semantic similarity metric based on the information content of the most specific class that subsumes the two words of interest in the taxonomy [28]. The information content of a concept C is $-\log P(C)$ where P is the probability that a randomly selected concept is a descendent of C. There are many variants of this basic idea. In [8], Lin proposes a similarity metric equal to the twice the information content of the most specific class that subsumes the two words of interest in the taxonomy divided by the sum of the information content of those two words. The rationale is that the similarity between two concepts is a function of their commonalities and differences. Alternatively, because the differences are equal to their complete description less their commonalities, we can say that the similarity of two words is a function of their commonalities and descriptions. The commonalities are captured by the most specific class that subsumes both concepts, while the descriptions are captured by the information content of the concepts themselves. Lin's metric had a correlation of .834 to human assessments (in contrast, when humans try to recreate the similarity assessments of other humans, correlation is .885 [9]). Other semantic similarity metrics are based on the distance between two words in a taxonomy such as WordNet. Additional semantic similarity measures are described by Widdows in the fourth chapter of his book "Geometry and Meaning" [10], and Budanitsky and Hirst compare the performance of five such metrics experimentally in [11]. The Java WordNet Similarity Library is an open source implementation of several semantic similarity metrics [14].

Lexical and semantic methods work well for comparing individual string values such as the names of entities, but they cannot be used alone to compare groups of items, such as the descendents of different entities. A set similarity metric is needed for this purpose. One common such metric is the Dice coefficient. The Dice coefficient of two sets is equal to twice the size of their intersection divided by the sum of their sizes. A very similar metric is the Jaccard index, which is equal to the size of the

intersection of two sets divided by the size of their union. As an example of set comparisons, in order to compare the descendents of two entities we generally use a combination of lexical and semantic methods to determine which of their descendents are equal, compute the intersection (and possibly union) of the two sets using this equivalence information, and then compute the Dice or Jaccard value (or other set similarity metric). It is also possible to recursively compute set metrics on these entities rather than just using lexical and semantic metrics to determine equality; however, this is computationally expensive and care must be taken to avoid infinite loops. Measures of graph isomorphism could also be considered a type of set similarity metric in the ontology mapping domain, but these are not widely used in current alignment algorithms. Set similarity metrics are often useful in comparing structural aspects of two entities, such as their ancestors, descendents, neighbors, etc.

2.4. Similarity Aggregation

Overall, most of the research done in the field of ontology mapping has been concerned with developing appropriate similarity metrics. Comparatively little research has been done with respect to similarity aggregation. The standard approach is to simply normalize the values produced by the similarity metrics and then add them, possibly weighting the individual metrics different amounts. The only variation the author is aware of is the work presented by Ehrig and Sure in [22] in which they compare the traditional weighted sum of linear inputs (where the weights were determined by an expert) to a weighted sum of sigmoid functions applied to each input and a weighted sum approach in which the weights were determined by a neural network. The sigmoid function performed the best in their experiments. Intuitively, this function focuses more on metrics whose values indicate similarity while mostly ignoring those that indicate no relation.

2.5. Interpretation

Interpretation is the process of taking the aggregated similarity metrics between all the different entity pairs considered and producing a set of (one-to-one, in our case) mappings between entities. There are a variety of ways to go about this, some of which depend on the choices made during the computation and aggregation stages.

When the value of a similarity metric is calculated, it alone does not tell us whether the two entities being compared should be considered equivalent – some type of threshold must typically be applied (e.g. “if the Levenstein distance is less than 3, then the entities are considered equivalent”). If several metrics are employed, the aggregation stage can normalize them and combine them into a single similarity value, or each metric can report a yes/no vote on the similarity of the two entities rather than the metric’s raw value. In either case, a threshold – either an overall raw similarity value or a number of yes votes (and threshold values for the individual metrics) – is needed for interpretation. Determining an appropriate value for this threshold is a difficult task.

Some systems designed to facilitate ontology mapping avoid the need to determine an appropriate threshold (and the whole interpretation question in general) by passing the decisions on to their users. Many ontology editors, such as the Concept-map Ontology Environment, allow users to open up two ontologies at once and manually generate mappings between them by adding explicit equivalence relations. Completely manual ontology mapping is practically infeasible for all but very small ontologies. Alternatively, PROMPT is a semi-autonomous ontology alignment tool that iteratively presents its user with suggested mappings and allows him or her to decide which mappings are valid [16]. PROMPT is available as a plug-in for the popular open source Protégé ontology editor [23]. Fully autonomous ontology mapping systems may provide a default similarity threshold and allow the user to modify it. It is also conceivable to use a heuristic or machine learning technique to arrive at a suitable threshold value, but the authors are not aware of such an approach.

In autonomous ontology mapping systems, once an appropriate similarity threshold value has been set there is still the issue of how to choose mappings when several pairs of entities exceed the threshold. Often several different mapping sets are equally valid and one must either choose randomly or impose additional constraints to differentiate among the available options. For instance, Qazvinian et al. use a genetic algorithm to optimize overall similarity and edge preservation [24]. In [25] Melnik et al. note that the problem of choosing a “good” mapping can be considered analogous to the stable marriage problem, well-known from graph theory. There are known algorithms for efficiently generating solutions to the stable marriage problem. The problem can also be considered similar to finding the maximum weighted matching, which has also been examined as part of graph theory for quite some time. Heß compared the performance of applying the stable marriage or

maximum-weight constraints on the chosen mappings and found that the maximum-weight approach performed significantly better on the OAEI dataset [26].

2.6. Alignment Debugging

Several of the more recently developed ontology mapping algorithms incorporate an alignment debugging phase. As mentioned in the previous section, there are often several reasonable mappings for a given entity and choices between them are frequently made incrementally using heuristics. In some cases, these individual choices may not be coherent when considered as a whole. For example, assume the mapping algorithm produces the partial interpretation shown in Figure 2. Because it is unusual (though not impossible) that two entities which are siblings in one ontology would map to two entities that have a parent-child relationship in a second ontology, the incongruous mappings might be marked for reconsideration during a future iteration.

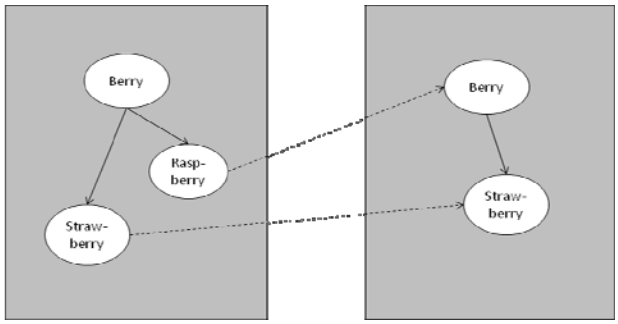

Figure 2. Alignment Debugging

ASMOV checks for five different types of inconsistencies during its alignment debugging phase, which are described more thoroughly in [2]. In essence, these checks assert that if two entities in the first ontology are related in some way then the entities that they are mapped to in the second ontology should be related in the same way. The specific relations checked are parent-child, disjointness, subsumption, and domain/range of properties applied to classes. In addition, the requirement of one-to-one mappings is verified. Lily [3] uses the debugging method described in [15], which involves locating mappings that are redundant, imprecise, inconsistent, or abnormal. Some of these checks involve mapping hierarchical relations other than equivalence; the checks for inconsistent and abnormal mappings are most useful for equivalence-only alignment algorithms. These are quite similar in reasoning to those done by ASMOV. The alignment debugging process may attempt to resolve the inconsistencies automatically or it may flag problematic mappings to be re-examined in later iterations. OMEN uses Bayesian networks on an ontology alignment to enhance existing mappings and invalidate false ones in a probabilistic approach to alignment debugging [20].

In practice, current ontology alignment systems are generally not accurate enough for most applications. Therefore, when alignment debugging is not done explicitly by the system (and often even when it is), the task of sanity-checking the mappings produced falls to humans.

3. TOM APPROACH

The rationale behind the Targeted Ontology Metric is that every pair of ontologies to be merged has different characteristics that are best dealt with by different combinations of similarity metrics. Our goal is to provide TOM with an array of the best aspects of current state-of-the-art ontology mapping algorithms and then allow it to dynamically choose which of those aspects to use based on an analysis of their performance on a subset of the problem at hand.

3.1. Overview

We now describe TOM according to the framework introduced in Section 2. The algorithm uses JENA to parse OWL ontologies written in RDFS and extracts a variety of features, described in more detail in Section 3.2. TOM has a variety of similarity metrics available to it, which are also described more fully in Section 3.2. The algorithm selects a subset of each ontology. Using these subsets together with the mappings that exist between them (provided by humans), TOM determines which of the available lexical and semantic metrics perform the best on this particular pair of ontologies. A genetic algorithm (described in more detail in Section 3.3) then selects the applicable features to be considered on-the-fly. The interpretation step looks at the vote tally of the different chosen similarity metrics. In the current implementation, the suitable values for the thresholds of the individual similarity metrics and the vote tally were arrived at through empirical experimentation and are not adjusted for different ontology pairs – this is a potential area of future research. The alignment debugging phase has not yet been implemented – we plan to use the technique described in [2]. The current TOM implementation is meant to test the general approach rather than run in a production system and therefore simply compares every entity in the first ontology to all of the entities of corresponding type (classes to classes,

relations to relations, etc) in the second. It is possible to add a next step heuristic (as described in Section 2.2) such as that used in *QOM* [5] in order to improve the speed of the algorithm. TOM iterates until no new mappings are found.

Table 1. Entity Features

Entity	Feature	Int	Ext	Lex	Sem	Com	Set
Class	Name	X		X	X		
	URI	X		X			
	Direct Instances		X			X	X
	All Instances		X			X	X
	Superclasses	X				X	X
	Subclasses		X			X	X
	Ancestors		X			X	X
	Descendents		X			X	X
Prop.	Name	X		X	X		
	URI	X		X			
	Domain	X				X	X
	Range	X				X	X
	Superprops	X				X	X
	Subprops		X			X	X
	Ancestors		X			X	X
	Descendents		X			X	X
Indiv.	Label	X		X			
	Data values	X		X			X

Table 2. Metrics

Type	Similarity Metric
Lexical	Levenshtein
	SMOA
	JaroWinkler
Semantic	Pirro-Seco
	Resnick
	Jiang
Set	DICE
	Jaccard

3.2. Features and Metrics

The features used by TOM are shown in Table 1. For each feature, the table explains to what type of ontological entity (class, property, or individual) the feature applies, whether the feature is intrinsic (local to the entity) or extrinsic (requires information outside the entity, such as knowledge of its neighbors), and what types of similarity metrics are applicable for comparing instances of this feature. When the feature represents a set rather than a single element (e.g. ancestors versus name) then either a lexical or semantic or a combination of the two (indicated by the composite column) will need to be used in conjunction with a set metric. Table 2 contains a list of the similarity metrics that TOM has available to it, categorized in a manner corresponding to the last three columns of Table 1. These metrics are defined in Section 2.3. Additional features and metrics can be added to the system in a modular fashion.

3.3. Genetic Algorithm

A genetic algorithm (GA) is a biologically-inspired search process [27]. A population of individuals is created prior to beginning the evolutionary process. There are several different strategies for building the initial population, but we use a straightforward approach: one individual is set to the lexical metric on the class and property names (a strategy that has been shown to be reasonably effective in many cases), while the genes in the remaining chromosomes are set to either 0 or 1 with a probability of .75 and .25 respectively. This is done to focus the search on solutions involving fewer metrics, in the interest of lowering the computation time for the composite metric. The set metric uses the best individual from the population as its composite equality metric. Each generation (or epoch), the individuals in the population reproduce. This is done by selecting two chromosomes (higher-performing individuals have a better chance of reproducing), randomly choosing a point along their length called a crossover point, and exchanging the genes from that point onward (for single-point crossover). Each chromosome also has a random chance of having one of its genes mutated (the value of the gene is flipped). Reproduction provides depth to the search in order to focus on promising areas, while mutation adds breadth to avoid prematurely converging on a local optimum within the search space. After the reproduction phase, the population is evaluated with respect to a fitness function, which is described below. Individuals are chosen to survive to the next generation using a deterministic tournament selection strategy with k=3. It is possible for the fitness of the population to decrease (e.g. if the best individual is mutated in a disadvantageous manner). In this case, the best individual is restored to the population. The algorithm iterates until a stopping criterion is reached, which in this case was chosen to be when the change in the fitness of the best individual between two generations falls below a threshold.

3.3.1. Chromosome Representation

The current implementation reduces the size of the search space by determining in isolation (prior to beginning the GA) which single metric in the lexical, semantic, and set groups performs best (class name and subclasses are used as test cases for this purpose). The chromosome then need only represent a choice between activating the feature using the applicable metric or ignoring the feature altogether. This results in a chromosome containing twenty binary genes (note that both lexical and semantic

metrics can be applied to class and property names). The size of the search space is therefore 2^{20}, which equals 1,048,576. A genetic algorithm should be able to search a space of this size reasonably efficiently. There is of course a drawback to making simplifying assumptions such as choosing a single lexical, semantic and set metric apriori. Deciding about the optimal metric of each type in isolation is a form of greedy search heuristic and ignores potential gains related to the interaction of different metrics/features. For instance, the effectiveness of a lexical metric may vary depending on the feature to which it is applied or on the choice of set metric.

3.3.2. Fitness Function

The fitness function of the GA is the most interesting aspect of the TOM approach. Prior to initiating the GA, we select a small subset of each ontology. There are several possible strategies to choosing this subset. We begin by choosing all of the entities that do not have a parent, along with the direct neighbors of those entities. The rationale is that these entities are likely to be the most general concepts within the domain and therefore are the most likely to have matches in a second ontology that describes the same domain. It is possible to include more than one level of neighbors in this selection – the number of entities selected from each ontology can be a parameter of TOM and should in practice be related to the overall size of the ontologies. Our selection algorithm also insures that there is a mix of both classes and properties. Investigating other subsection selection strategies is a potential area of future work. After the ontology subsets have been established, the GA begins. For each individual chromosome in the population, the active similarity metrics are computed for each pair of entities in the ontology subsets. Each metric "votes" on whether or not the pair of entities is equivalent, and the overall vote tally is counted. TOM is envisioned to eventually run in both a supervised and an unsupervised mode (comparable to semi-autonomous versus autonomous approaches). In supervised mode, which is the focus of this paper, the user provides the appropriate mappings between the two ontology subsets, and the fitness of each individual is judged with respect to f-measure (defined in Section 4.2) based on this ground truth along with a penalty factor based on the number of different metrics used, in order to give preference to simpler solutions.

3.4. Related Work

Several researchers have applied genetic algorithms or other machine learning-based approaches to the ontology mapping domain, but most of this work has been with respect to similarity computation rather than choosing which metrics to use and how they should be aggregated [18]. One work that does attempt to tackle this latter issue is the GOAL algorithm by Martinez-Gil and his colleagues [17]. Their system attempts to choose which of four similarity metrics (Levenstein, SIFO, Stolios, and QGrams). The fitness function allows the user to optimize one of precision, recall, f-measure, or false positives. While the authors do not state this directly, it is evident from these options for the fitness function that GOAL optimizes the choice of metrics *only when the correct answers are already known*. No mention is made of using a subset of the ontology as a training set, which implies that the user must supply GOAL with the complete mapping for a pair of ontologies. This does not appear to have much practical value, except possibly if many ontology pairs with similar characteristics to the training pair must be aligned.

As mentioned in Section 2.4, Ehrig and Sure attempted to use a neural network to determine the appropriate weights for the similarity metrics used in their system. They used 20% of the entities in the ontologies as a training set and employed a neural network consisting of a linear input layer, a hidden layer with a tanh function, and a sigmoid output layer. The resulting weights did not perform as well as the authors expected. They attributed this to over-fitting of the data and suggested that the general approach had merit [22]. A GA approach that attempts to determine only which metrics to include and not their weights may be less prone to over-fitting. The other difference between Ehrig and Sure's work and TOM is that they used several similarity metrics that were specific to the knowledge domain they were considering.

GLUE is another machine learning approach to ontology mapping. It is based on the joint probability distribution between pairs of entities. Two base classifiers are learned in order to compute these probabilities, and a meta-classifier is learned to aggregate their results. While the authors are somewhat unclear on this point, it appears that GLUE requires a fairly extensive amount of instance data for training the classifiers (30-90 instances per node) and has a relatively high computational complexity as both the base classifiers and the meta-classifier need to be trained [19]. In general, machine learning based approaches to different aspects of the ontology matching problem often involve text processing of documents semantically related to the entities in the ontologies. Both GLUE and OMEN (discussed in Section 2.6) fall into this category, as does the approach described by Pan et al. in

[21]. GOAL and Ehrig and Sure's neural network approach are exceptions.

4. EXPERIMENTAL SETUP AND RESULTS

In this section we describe the dataset used throughout this research effort and some preliminary results of the genetic algorithm used to determine which entity feature and similarity metric combinations will be considered by TOM.

4.1. OAEI Contest

The Ontology Alignment Evaluation Initiative [1] was started in 2004 with the goal of making it easier for researchers to compare the results of their ontology matching algorithms. The organizers hold a contest each year in which participants run their algorithms on a large set of ontology matching problems and compare the results based on precision, recall, and f-measure.

The OAEI suite of tests contains five tracks that test various aspects of ontology mapping algorithms, including their performance on relations other than equivalence, very large ontologies, and as part of a larger application task. There are both synthetic and real world ontologies of varying size and complexity, and the results are either available to the researcher during the test, withheld until after the test, or evaluated individually by a human expert. We focus here on the benchmark track, which consists of 111 pairs of variations on a bibliographic ontology. The base ontology has 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. The base ontology is systematically altered in different ways in order to test different aspects of the ontology matching problem. Each test involves comparing the original ontology with one of its mutations. The mutations are based on different combinations of six types of modification (as described in [1]):

- Name – entity names can be replaced by random strings (R/N), synonyms (S), names with different conventions (C), or translated to a foreign language (F)
- Comments – comments can be removed (N) or translated to a foreign language (F)
- Specialization Hierarchy – the hierarchy information can be removed (N), expanded (E), or flattened (F)
- Instances – can be removed (N)
- Properties – the restrictions (R) on classes can be removed or the properties can be removed entirely (N)
- Classes – can be expanded (E) (i.e. replaced by several classes) or flattened (F)

4.2. Preliminary Results

The genetic algorithm was run with a population size of thirty, a mutation rate of 0.05 (for each gene), and a stopping criteria of 0.005 (i.e. when the fitness of the best individual failed to increase by that amount, the algorithm halted). The ontology subsets were twenty percent of the full ontology sizes. Seven cases from the OAEI benchmark suite were used for this initial test – they were numbers 201, 205, 206, 222, 224, 228, and 301. Because GAs are nondeterministic each test case was run five times. The results are shown in Table 3, which contains the test identification number, the best-performing chromosome, the average over the five runs of the f-measure on the ontology subsets, and the average f-measure on the full ontology mapping problem. The genes correspond to the list in Table 1.

Table 3. Preliminary Results

Test	Best-performing Genome	Avg Subset F-measure	Avg Full F-measure
201	01000001000001000000	0.2286	0.0588
205	00001000110000010000	0.7101	0.2953
206	10000000010010011000	0.4152	0.4936
222	10000000010000000000	0.9143	0.6567
224	10000000010000000000	0.9333	0.7564
228	10000000010000000000	1.0000	0.8303
301	10000000010000000010	0.7566	0.6645

As expected, the f-measure on the subsets was most often higher than that of the f-measure on the complete ontologies. However, these two values were correlated enough to enable the genetic algorithm to tailor a metric set that was relevant to the particular problem at hand.

201 is the most difficult of the test cases considered. The class names were removed. The best metric set found consisted of the semantic class name, ancestor classes, and property ranges. Because class names were removed, the first of these metrics is not capable of contributing to the overall f-measure. The performance on this test was poor.

Test 205 replaced class names with synonyms. The best metric set found involved all of a class's instances and

descendent classes, along with lexical comparison of property names and subproperties. This is a reasonable metric set, although it could be argued that semantic comparison of class names would have improved accuracy.

Test 206 translated the class names from English to French. The metric set that was evolved used lexical comparison of class names, lexical comparison of property names, property domains, subproperties, and ancestors. This is again a reasonable choice of metrics. In particular, a lexical rather than semantic comparison of class names is more likely to result in accurate matches in this situation.

Test 222 flattened the hierarchy (removed class-subclass relationships), test 224 removed all instances, and test 228 removed all properties. The metric set found in all three of these cases was lexical comparison of class and property names. This is completely reasonable in the first two tests, and reasonable but inefficient in the third test (because the property metric will be ignored due to the lack of properties).

Finally, test 301 compared the reference ontology to the real-world BibTex ontology from MIT. The evolved composite metric consisted of lexical comparison of class names, property names, and instance labels.

In summary, while the overall f-measures are not competitive with the OAEI competitors from 2009 (which used unsupervised algorithms), fine-tuning the algorithm parameters, particularly the metric thresholds, will likely improve performance drastically. Meanwhile, these preliminary results do indicate that a GA can construct composite metrics that take advantage of the characteristics particular to the problem at hand. The author wishes to stress the preliminary nature of these results – much more testing needs to be done to establish the validity of this approach.

5. CONCLUSIONS AND FUTURE WORK

Ontology mapping is a hard problem – even humans have a difficult time aligning their internal mental models with one another – but it is one that needs to be solved if ontologies are going to be used to provide semantic interoperability among systems. Current approaches to the ontology mapping problem are not suitable for most real-world applications in terms of accuracy, scalability, or response time. One of the goals of this paper was to provide system designers with an idea of the current state of ontology mapping research and the gaps that need to be addressed. These issues are summarized here in the form of a list of questions to consider before relying on an ontology-based approach to system integration.

- What ontology matching algorithm will be used and how does its performance compare with others in terms of both accuracy (precision, recall, f-measure) and memory and computation requirements?
- What is the size of the largest pair of ontologies that can be processed?
- What parameters/thresholds does the algorithm involve and how can the appropriate values be determined?
- Does the algorithm do one-to-one matching or can it handle other types of relations?
- What does the algorithm require from the user – parameter values, thresholds, sample mappings, interpretation/verification of results, etc?

This paper also introduced a new ontology mapping algorithm – Targeted Ontology Mapping – specifically designed to perform at a consistent level on a wide variety of ontology pairs. TOM uses a genetic algorithm on a carefully selected subset of the ontologies being aligned to dynamically determine which set of similarity measures provide the best accuracy for the least cost. Preliminary results show that the GA is capable of finding an appropriate combination of features and metrics in a reasonable amount of time.

Future work will involve completing the implementation of TOM and finding more optimal values for parameters such as metric thresholds, population size, and stopping criterion. We will also consider the possibility of an unsupervised version of TOM.

REFERENCES

[1] Ontology Alignment Evaluation Intiative, http://oaei.ontologymatching.org/.

[2] Y. Jean-Mary, E. Shironoshita, and M. Kabuka. “ASMOV: Results for OAEI 2009,” Fourth International Workshop on Ontology Matching, Washington D.C., USA, 2009.

[3] P. Wang and B. Xu. “Lily: Ontology Alignment Results for OAEI 2009,” Fourth International Workshop on Ontology Matching, Washington D.C., USA, 2009.

[4] P. Shvaiko and J. Euzenat. "Ten Challenges for Ontology Matching," Seventh International Conference on Ontologies, Databases, and Applications of Semantics, 2008.

[5] M. Ehrig and S. Staab. "QOM – Quick Ontology Mapping," Lecture Notes in Computer Science, vol. 3298, pp. 683-697, 2004.

[6] G. Stoilos, G. Stamou, and S. Kollias. "A String Metric For Ontology Alignment," ISWC 2005, Lecture Notes in Computer Science, vol. 3729, pp. 624-637, 2005.

[7] W. Winkler. "The state record linkage and current research problems," Technical report, Statistics of Income Division, Internal Revenue Service Publication, 1999.

[8] D. Lin. "An Information-Theoretic Definition of Similarity," Fifteenth International Conference on Machine Learning, pp. 296-304, 1998.

[9] J. Jiang and D. Conrath. "Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy," International Conference Research on Computational Linguistics (ROCLING X), 1997, Taiwan.

[10] D. Widdows. GEOMETRY AND MEANING, CSLI publications, 2004.

[11] A. Budanitsky and G. Hirst. "Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures," Workshop on WordNet and Other Lexical Resources, 2001.

[12] "Jena – A Semantic Web Framework for Java," http://jena.sourceforge.net/index.html.

[13] The SecondString library, http://secondstring.sourceforge.net/.

[14] G. Pirro and N. Seco. "Design, Implementation and Evaluation of a New Similarity Metric Combining Feature and Intrinsic Information Content," ODBASE 2008, LCNS, Springer Verlag, 2008.

[15] P. Wang and B. Xu. "Debugging Ontology Mappings: A Static Approach," *Computing and Informatics*, vol. 22, 2003, 1001-1015, v 2007-Feb-14.

[16] N. Noy and M. Musen. "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," 17th National Conference on Artificial Intelligence (AAAI'02), Edmonton, Alberta, Canada, August 2002.

[17] J. Martinez-Gil, E. Alba, and J.F. Aldana-Montes. "Optimizing Ontology Alignments by Using Genetic Algorithms"

[18] J. Wang, Z. Ding, and C. Jiang. "GAOM: Genetic Algorithm based Ontology Matching," 2006 Asia-Pacific Conference on Services Computing (APSCC'06), 2006.

[19] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. "Ontology Matching: A Machine Learning Approach," HANDBOOK ON ONTOLOGIES, edited by S. Staab and R. Struder, Springer-Verlag, 2004.

[20] P. Mitra, N. Noy, andA. Jaiswal. "Ontology Mapping Discovery with Uncertainty," In Gil, Y., Motta, E., Benjamin, V.R., Musen, M.(Eds.) Fourth International Semantic Web Conference (ISWC), Galway, Ireland, *Lecture Notes in Computer Science*, Springer Verlag GmbH, 3729 (2005): 537-547.

[21] R. Pan, Z. Ding, Y. Yu, and Y. Peng. "A Bayesian Network Approach to Ontology Mapping," *Lecture Notes in Computer Science*, Springer, vol. 3729, 2005, pp. 563-577.

[22] M. Ehrig and Y. Sure. "Ontology Mapping – An Integrated Approach," *Lecture Notes in Computer Science*, Springer, vol. 3053, 2004, pp. 76-91.

[23] The Protégé Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu/.

[24] V. Qazvinian, H. Abolhassani, and S. Hossein Haeri. "Coincidence Based Mapping Extraction With Genetic Algorithms," Third International Conference on Web Information Systems and Technologies (Webist 2007), Barcelona, Spain, March 2007.

[25] S. Melnik, H. Garcia-Molina, and E. Rahm. "Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching," 18th International Conference on Data Engineering (ICDE'02), San Jose, California, February, 2002.

[26] A. Heß. "An Iterative Algorithm for Ontology Mapping Capable of Using Training Data," *Lecture Notes in Computer Science*, Springer, vol. 4011, 2006, pp. 19-33.

[27] J.H. Holland. ADAPTATION IN NATURAL AND ARTIFICIAL SYSTEMS, Ann Arbor: The University of Michigan Press, 1975.

[28] P. Resnik. "Using information content to evaluate semantic similarity in a taxonomy," 14th International Joint Conference on Artificial Intelligence, pp. 448-453, 1995.

[29] M.A. Jaro. "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida," *Journal of the American Statistical Association*, vol. 89, pp. 414-420.