Correspondence Pattern Attribute Selection for Consumption of Federated Data Sources

Brian Walshe[1], Rob Brennan[1], Declan O'Sullivan[1]

[1]FAME and Knowledge and Data Engineering Group, School of Computer Science and Statistics
Trinity College Dublin
Dublin, Ireland
{walshebr, rob.brennan, declan.osullivan}@scss.tcd.ie

Abstract—**When consuming data from federated domains, it is often necessary to identify the relationships that exist between the data schemas used in each domain. Discovering the exact nature of these relationships is difficult due to data set schema heterogeneity. Prior work has focused on inter-domain class equivalence. However it is not always possible to find an equivalent class in both schemas. For example, when instances are modeled as classes in one domain (e.g. router type) but as the attribute values of a single class in the other domain (e.g. router interface). This paper investigates whether when classifying instances in one data set against a second schema, it may be more useful to use some attribute (or attribute group) other than the original class type, to perform this classification. A machine-learning based classification approach to appropriate attribute selection is presented and its operation is evaluated using two large data-sets available on the web as Linked Data. The classification problem is compounded by the less formal semantics of Linked Data when compared to full ontologies but this also highlights the strength of our approach to dealing with noisy or under-specified data-sets and schemas. The experimental results show that our attribute selection approach is capable of discovering appropriate mappings for cases where the correspondence is conditioned on one attribute and that information gain provides a suitable scoring function for selection of correspondence patterns to describe these complex attribute-based mappings.**

Keywords—Data Federation, Attribute Selection, Semantic Mapping

I. INTRODUCTION

Sharing data is of fundamental importance to federated domains, whether directly via access to federated data-bases or triple-stores or indirectly through service and query interfaces. However, despite the ubiquity of the Internet as a generic data sharing infrastructure, the main obstacle to federated data sharing remains data schema (structure) diversity and opaqueness.

In recent years ontology or knowledge-model-based approaches to data or knowledge modeling and publication have provided a standards-based[1] means to capture the semantics or meaning behind data and to make the schemas queryable. This addresses the federated data sharing challenge of opaqueness; however the challenge of data diversity remains. Even with explicitly defined semantics, different people's viewpoints and the level of detail they use in describing a problem domain mean that while the ontologies give a better starting point than, for example, a database schema, it is still necessary to work to resolve the differing views of the data. If one ontology, O_1, provides a rich class hierarchy to describe network components with separate classes for managed and unmanaged network switches, while another ontology, O_2, uses a shallower hierarchy with only a single class for describing switches, how should we describe the relationship between a managed network switch in O_1 and a network switch in O_2? Should we simply say that the types are similar? Is it possible to use some other information to make the relationship more explicit? For example, if instances of type NetworkSwitch in O_2 have attributes describing their SNMP details, might we infer their class is equivalent to ManagedNetworkSwitch in O_1?

Finding and describing the relationship between ontology elements is known as Ontology Mapping. Within this there are two main tasks, first the *matching process* which involves finding the concepts that are related, and then the *mapping process* which describes how these concepts are related. Ontology Mapping is an active topic, with the matching process being the most investigated component of the field [1].

Discovering and describing the relationship between the semantically related elements in independent sources is a relatively old problem. It can be found in the field of database integration, where many common forms of heterogeneity have been identified [2][3] which impede our ability to easily combine data found in federated databases. Classical approaches to ontology mapping have emphasized finding relationships between classes in the source and target ontologies. Due to the differing ways that ontologies are modeled this may not always provide a suitably expressive method for describing mappings, and more complex forms of mappings may be necessary, such as mappings between subsets of named classes defined by the attributes rather than whole class-to-class mappings [4]. This means that in some cases sub-sets of the instances of a class in the source domain are mapped to a more specialized class in the target domain. In recent work [4], Scharffe and Fensel call these types of complex mappings "correspondence patterns".

A relatively new area where the importance of these correspondence patterns can be demonstrated is for the consumption of Linked Data [5], which is loosely structured RDF published on the web. Typically such data has many

[1] Most notably the W3C's Semantic Web suite of standards including Resource Description Framework (RDF), the Web Ontology Language OWL and SPARQL for querying RDF.

978-1-4673-0269-2/12/$31.00 ©2012 IEEE

inconsistencies, both internally and with respect to its schema, and is not as formally described as a pure OWL ontology – perhaps using RDFS (RDF Schema) to define a number of component vocabularies. Such data is often unequipped with a comprehensive class hierarchy for use by classical mapping methods, and the presence of errors means that correspondence patterns can be used to clean up data for consumption.

In such a system we wish to enable the automated or semi-automated detection of appropriate transformational correspondences rather than the old focus on concept matching between rich and formally described ontologies. A key focus of such an automated technique is the identification of instance attributes or attribute values that can be used to define class membership in the target schema (usually the internal schema of the Federation member). Hence this defines the attributes used and the specific correspondence pattern form in constructing a complex mapping.

Given two overlapping, error-containing, semi-formally described data sets to be mapped, we have the following research questions:

- What is a suitable algorithm to identify the appropriate attributes and correspondence patterns to enable automated generation of complex mappings between instances in the source data-set and classes in the destination data-set schema?
- How much, if any, training data or background knowledge is required by such an algorithm?
- What is a suitable strategy to deal with arbitrary ranges of attribute values in a generic algorithm?
- How would such an algorithm fit into an overall mapping framework or process?
- What accuracy could be obtained by such an algorithm?

The work presented in this paper addresses the first of these research questions. In this paper we describe a novel approach to this problem based on the idea that, given limited training data, a machine learning attribute selection algorithm can use the instance data in two data-sets to infer the complex mappings between those data-sets. More specifically, our algorithm applies a standard information gain measure to identify the particular attributes or attribute values to be used for constructing complex mappings as a correspondence pattern. This provides the basis for a tool which would generate candidate executable correspondence patterns (complex mappings) between two data-sets, even in the presence of data-set errors and limited availability of full, formal schemas or ontology definitions describing the data. In order to evaluate our method we have carried out an experiment on two very large linked data sets describing people in the movie/entertainment domain – dbpedia [6] and YAGO [7].

The rest of this paper is laid out as follows: section II presents related work, section III a formal problem statement, section IV proposes our mapping algorithm approach, section V describes an experiment that we have performed to evaluate our approach, and finally section VI presents conclusions and future work.

II. BACKGROUND

1) Data Sets

RDF/Linked data is an accessible source of real life instance data developed independently and published on the web. This data is well described and intended for ease of integration, yet this is still not a straightforward task. A primary reason for the difficulty in integrating these sources is the differences in the way they were designed. An example of this are the YAGO [7] and DBpedia [6] projects. Both of these projects provide linked data sources generated form Wikipedia[2] entries, and we would expect them to be extremely similar – yet actually their structure is quite different.

YAGO provides a rich class hierarchy based on WordNet [8]. In YAGO, a class is a sublclass of another one, if the first's set of synonyms is a hyponym of the second's. YAGO allows very specific classifications – such as *American people in Japan* – it contains 149,162 classes, and 143,210 subClassOf declarations. In contrast, the DBpedia Ontology classification scheme consists of 170 classes that form a shallow subsumption hierarchy.

2) Ontology Mapping

Ontology Mapping tools commonly use lexical similarity, graph comparison, or a combination of these techniques to produce matches and mappings. The S-Match [9] tool, for example, treats the ontologies to be matched as labeled graphs. It uses background information such as WordNet to remove ambiguities such as homonyms and synonyms from the graphs' labels. The semantic meaning of the nodes is inferred by first looking at the meaning of their labels (without considering context) and then refining this meaning by considering the node's position in the schema tree. Once semantic meaning for each node in each graph has been evaluated and refined, matching the nodes in each tree can be solved as a standard Boolean satisfiability problem.

S-Match is capable of finding the relationships: equivalence ($\equiv$), less general ($\sqsubseteq$), more general ($\sqsupseteq$) and disjointness ($\not\equiv$), between the concepts in the ontologies. This is common of most ontology alignment tools – in general they map concepts based on their *rdf:type* attribute, with equivalence being the most common type of mapping. Due to the differing ways that ontologies are modeled this may always not provide a suitably expressive method for describing alignments, and more complicated forms of mappings may be necessary. For example these relationships cannot accurately model the relationship $\text{o1:NetworkSwitch}|_{\text{o1:hasSNMPAgent=x}} \rightarrow \text{o2:ManagedNetworkSwitch}$, as described in section 1.

These types of complex mappings are called Correspondence Patterns in [4]. They cover a wide range of cases, including not only the more standard relationships such as equivalence and subsumption, but also *Conditional* and *Transformation* patterns. Conditional patterns are used when the scope of one

[2] http://wikipedia.org

entity in an ontology needs to be narrowed to match the scope of an entity in the other ontology, as in the network switch example presented in section 1. Possible conditions include constraints on the value of an attribute, its type, or simply its occurrence. Transformation correspondences include cases where elements must be altered in some way. For example this could include concatenating a first and second name together, or converting from one currency to another.

Current research in the area of Correspondence Patterns is focused on identifying and cataloging the different forms of pattern that exist. The EDOAL language [10] provides a method for describing complex forms of alignments. EDOAL uses an OWL-like syntax, though it is more rule orientated, allowing variables in expressions which can be used to describe constraints and transformations. It also contains additional constructs for expressing data transformations.

There are few matchers capable of generating the complex mappings that EDOAL allows, and support for the language in the Alignment API is still under development [10]. Ritze et. al. [11] describe a first attempt at a process for detecting complex mappings. It requires a set of input mappings which it refines using pattern matching at the schema level. A more machine learning approach is outlined in [12], which describes how the mapping task can be reformulated as an Inductive Logic Programming problem. They note however that there are many issues that must be resolved before this can provide a robust solution, issues such as incompleteness, inconsistency and uncertainty.

As stated previously, most ontology alignment techniques focus on analyzing schema. Typically they are intended to be used with well defined ontologies. Linked Data sources are not always as well defined, which may hamper schema based approaches. In [13] it was demonstrated that real world RDF vocabularies are often either over or under specified. *Overspecification* occurs when a class has many properties specified in the schema, yet real-world instances of the class rarely use these properties. A class is *underspecified* if instances of the class have many properties that were not specified in the schema. As real-world data may not adhere to its schema, this is something that we may wish to take into account when selecting appropriate correspondence patterns. If for example there is a choice of two patterns that are semantically equivalent it would be preferable to select the one that uses the more reliably set attributes in the instance data.

In summary, while the use of description languages such as RDF and OWL have made data sources more understandable, the diversity in the way they describe their data is still a challenge to federation. Even when they cover over-lapping domains, real world data sources often use very different schemata, as can be seen in YAGO and DBpedia. Current semantic mapping tools focus on describing simple relationships such as equivalence, but often we need more complicated relationships with rules or conditions. These more complicated mappings are still in their infancy, and most work has concentrated on providing methods for describing and sharing the mappings. Current initial attempts at automatically identifying complex mappings concentrate on pattern matching at the schema level. Real world semantic web data has been demonstrated to often deviate from its schema, so an approach that considers instance data may be of benefit when deciding on how best to specify a complex mapping.

III. Problem Statement

Differing conceptual models used in ontologies mean that simple class-class mappings are not always sufficient to describe the relationship between the elements of the ontologies. Some relationships are best described using a correspondence pattern that places some condition on an attribute of the instances of the class being mapped. Finding which condition should be used is a challenge. Schema information does not always tell us enough to make this choice, so instead we look to using instance level data.

Given a source ontology O_s and a target ontology O_t, a subsumption mapping between classes C_s and C_t – with $C_s \in O_s$ and $C_t \in O_t$ – and a set of instances, I, that exist in both O_s and O_t that each are members of both C_s and C_t, we wish to use the information we learn from I to refine the mapping from a subsumption relationship:

$$C_t \sqsubseteq C_s$$

to a Conditional Class Correspondence pattern of the form:

$$C_t \equiv C_s \mid_{a^*=v}$$

That is, all instances of class C_s with attribute a^* set to some value v are instances of class C_t. In addition to considering a^* equal to a specific value of v, we can also consider a^* set to any value, or type(a^*) = t, for some type t.

IV. Approach

Our approach relies on analyzing a user supplied sample of mapped instances to refine our understanding of the relationship between the classes of the instances found in two ontologies. In the scope of this paper, we make no attempt to infer any information from class labels or the structure of the ontologies taxonomies. Our process instead considers the structure *as used by the instance data*. There are several advantages to this approach, in that it allows us to consider attributes other than rdf:type contributing to an instance's classification. It does not require the schema to provide information such as the domain and range of attributes that would be necessary for a pattern matching based approach, as described in [11]. The main disadvantage of this approach however, is that it does require a set of pre-matched instances of the classes in the mapping to be refined.

To refine the subsumption mapping $C_t \sqsubseteq C_s$ where C_t is a class in our target ontology and is more specific than C_s the class in our source ontology it has been mapped to, we wish to find an attribute of the instances of class C_s which we can use to test if that instance is also a member of class C_t. We use the following process (as illustrated in fig. 1):

The user decides on the mapping to be refined $C_t \sqsubseteq C_s$

1. Using a set of pre-matched instances of our source and target ontologies, the user selects a sample of instances that have type C_t, and a set of instances that have type C_s, but not type C_t
2. These sets are combined to create our training sample and passed to the Attribute Ranker
3. The Attribute Ranker searches the source ontology to find all attributes the instances with type C_t in the training sample have been assigned.
4. Each of these attributes is ranked using a scoring function which evaluates the attributes ability to differentiate instances of types C_t and C_s

Constructing a suitable training set attribute selection is complicated by the fact that it is possible for each instnace to have multiple values for each attribute. Therefore, for each instance we must construct a bit vector, with one bit for each attribute and value combination. As this can lead to an extremely large number of attribute/value pairs to test, our method requires the user to manually pick which attributes to test. The scoring function can then evaluate which value of which attribute provides the best condition for refining the mapping. Future implementations will need methods for narrowing down the range of values considered for each attribute.

To refine the mapping $C_t \sqsubseteq C_s$, we require that our training set contain two kinds of instances, one set of positive matches $I^+ \subset C_t \cap C_s$, which are identified to have both class C_t and C_s and a set of negative matches, $I^- \subset C_s \setminus C_t$. Using this information, we can detect the existence of two kinds of Correspondence pattern, *Class by Attribute Value* – where our mapping is conditioned on a specific value of an attribute – and *Class by Attribute Existence* – where the specific value of the attribute does not matter, we only care if it has been set or not.

To evaluate Class by Attribute Value, the attribute to be tested, a^* must be specified. We then enumerate all possible values of a^* for the elements of I^+ and which we label $v_{1..n}$ We then construct indicator vectors for each element of $I^+ \cup I^-$ where each vector has its i^{th} element set to *1* if its corresponding instance has $a^* = v_i$ and *0* otherwise. Having constructed the indicator vectors we rank $v_{1..n}$ using their information gain score to find the most suitable value for the given attribute to condition the correspondence pattern on.

The Class by Attribute Existence pattern follows a similar procedure. Again, we use the sets I^+ and I^- defined as above, but no attribute to be tested is specified. In this case we enumerate all possible attributes that the instances of I^+ can have, and label them $a_{1..n}$. Our indicator vectors are constructed so that each vector has its i^{th} element set to *1* if the corresponding instance has a_i set to any value, and *0* otherwise. Again these are ranked to find the most suitable attribute, regardless of its specific value, to condition the correspondence pattern on.

There are several different metrics that could be used to provide our attribute rankings. One of the most common functions to use is an entropy measure based score known as

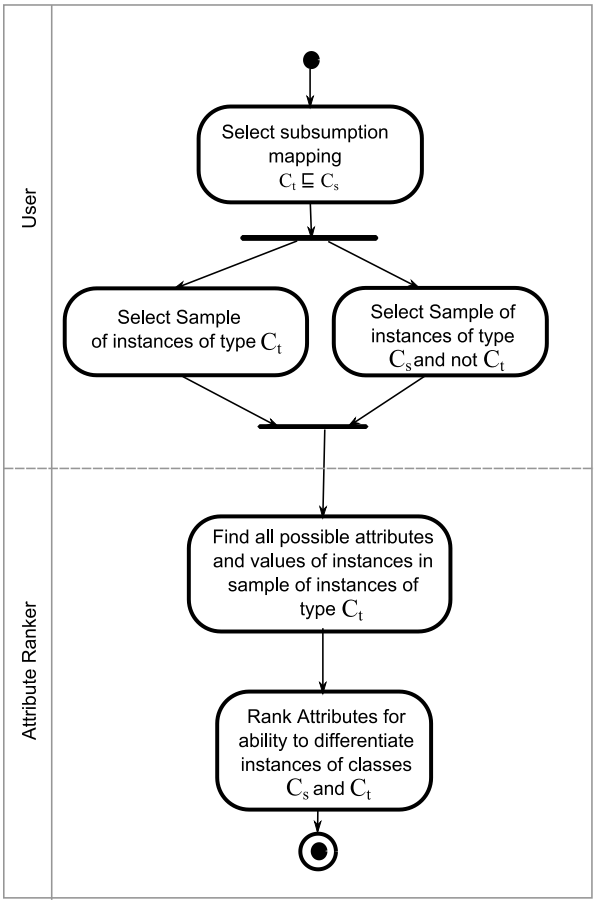

Fig. 1: The sequence for selecting an appropriate attribute to condition a correspondence pattern on. First the user creates a list of test instances, then the Attribute Ranker evaluates which attributes these instances may have, and how those attributes contribute to classifying the instances.

Information Gain (IG) [14], which we have elected to use. The χ-squared function was also considered, but this function can behave erratically for small sample sizes or low expected counts for individual classes. For each attribute, A, Information Gain measures the average decrease in the number of bits required to describe the class, C, of an instance if the value of A is known. It is defined as follows:

$$IG(C|A) = H(C) - H(C|A)$$

Where $H(C)$ is the entropy of C – the number of bits, on average, required to transmit a stream of values drawn from C's distribution:

$$H(C) = -\sum_c p(c) \log_2 p(c)$$

And $H(C|A)$ is the conditional entropy of C given A

$$H(C|A) = \sum_a p(a) H(C|A = a)$$

$$= \sum_a p(a) \left(\sum_c p(c|a) \log_2 p(c|a) \right)$$

Using the standard implementation of IG as our scoring function requires us to make a *closed world assumption* – that any statement that is not explicitly said to be true is false. This is in contrast to the *open world assumption* used in the semantic web, which requires statements to be considered as simply unknown if they are not explicitly said to be true or false. In addition, most implementations of IG expect each attribute to only have one value per instance. Because of this we need to perform an evaluation to test the suitability of IG as an attribute scoring function.

V. Evaluation

This section describes an experiment to evaluate the hypothesis that the Information Gain measure is an appropriate scoring function for selecting a correspondence pattern to refine a given semantic mapping. To evaluate this, the measure was used to select appropriate correspondence patterns for a set of initial semantic mappings. For each initial mapping we measured the information gain score for all possible correspondence patterns that can exist between the classes in the mappings. This was repeated for 100 randomly chosen training sets to test the mean and variance of the IG score for each attribute as well as the stability of the ranking, given different training sets. Ranking stability was measured by both the percentage of times a gold standard correspondence pattern was ranked first and the percentage of times it appears in the top five rankings.

We use the following process to establish these values:

1. For each mapping, M_i, we randomly generate 100 training sets.
2. For each training set, we record the IG score for all possible correspondence patterns for M_i.
3. We then rank the correspondence patterns and record if the gold standard pattern was the highest ranked or if it was in the top five.

Four initial mappings were used, based on YAGO and DBpedia. For each mapping a gold standard correspondence pattern was created, consisting of a condition on an attribute of the DBpedia class which narrows its scope to the more specific YAGO class. Searching the entire space of attributes and their possible values would be too great a challenge for this initial experiment, so the search was narrowed to three types of correspondence pattern:

- Class by Attribute Value correspondences conditioned on values of the rdf:type attribute
- Class by Attribute Value correspondences conditioned on values of the dbpedia-owl:occupation attribute,
- Class by Attribute Existence correspondences conditioned on the existence of any attribute.

The following initial mappings were considered.

M_1: yago:Actor ⊑ dbpedia-owl:Person

M_2: yago:Director ⊑ dbpedia-owl:Person

M_3: yago:Musician ⊑ dbpedia-owl:Person

M_4: yago:Politician ⊑ dbpedia-owl:Person

These particular relationships were selected as they allow us to easily select an appropriate gold standard correspondence pattern and they provide sufficient instances to allow us to create random sets of instance mappings which can be used to evaluate the IG scores of the correspondence patterns. When selecting the gold standards, preference was given to patterns conditioned on rdf:type if a suitable type was available. If not, occupation was chosen if this was reliably set in the instance data. If not an attribute existence pattern was used instead as the gold standard.

For Mapping 1, the condition that a person instance with rdf:type set to dbpedia-owl:Actor was selected as the gold standard. Mapping 2 is more problematic than that for actors, as there is no class in the dbpedia schema that corresponds to yago:Director. Instead the choice is to either select instances that have their dbpedia-owl:occupation set to dbpedia:Film_director or select instances that have been identified to have directed a film using the dbpedia-owl:director attribute. As the dbpedia-owl:occupation is set incorrectly for many instances, we elected to use the dbpedia-owl:director existence condition as our gold standard. For mapping 3 the condition rdf:type = dbpedia:MusicalArtist was used, and for Mapping 4 the condition dbpedia-owl:occupation = dbpedia-owl:Politician was used.

These patterns, expressed as conditions on attributes were as follows:

p_1: rdf:type = dbpedia-owl:Actor

p_2: dbpedia-owl:director = [any value]

p_3: rdf:type = dbpedia:MusicalArtist

p_4: dbpedia-owl:occupation = dbpedia:Politician

Testing the variance of the IG scores depending on the training set used, and the subsequent effect this has on the correspondence patterns' rankings required us to automatically generate sample training sets. We created these sets to have 30 examples instance mappings with the assumption that it in a real-world situation could be possible to manually discover this many examples. To create the these samples we used a

TABLE I. The number of times the gold standard was ranked first, and in the top 5.

Mapping	Gold Standard rank = 1st	Gold Standard ranked > 5th
M_1	*99*	*100*
M_2	*72*	*99*
M_3	*14*	*98*
M_4:	*44*	*89*

SPARQL SELECT query to retrieve all instances of the YAGO type being evaluated, then sampled randomly from this list.

Results:

Running the test for each of the mappings showed good results for mappings M_1 and M_2, ranking the gold standard correspondence pattern in first place, 99% and 72% of the time respectively. Correspondence pattern selection for M_3 and M_4 was less satisfactory with the gold standard correspondence pattern only being selected in first position 14% and 44% of the time respectively. Still, even for the mappings that our selection process performed poorly on, the correct correspondence pattern was reliably able to select the correct correspondence in the top five – 89% of the time for the worst case, M_3. These results are summarized in table 1, and are discussed in more depth below.

Mapping M_1: The correspondence pattern for this mapping is very straight forward, as dbpedia:Actor and yago:Actor are semantically similar. As such we would expect this correspondence pattern to be easy to detect. The mean and standard deviation information gain scores for the top five correspondence patterns refining mapping M_1 are displayed in fig. 2, where rdf:type = dbpedia-owl:Actor can be seen to score much higher than the other attribute/value pairs.

Mapping M_2: The correspondence pattern for this mapping is more difficult as there is no class in DBpedia that corresponds to yago:Director. While this did have the highest mean IG score, the pattern rdf:type = dbpedia-owl:Actor also scored highly, which can be explained by the fact that many directors are also actors.

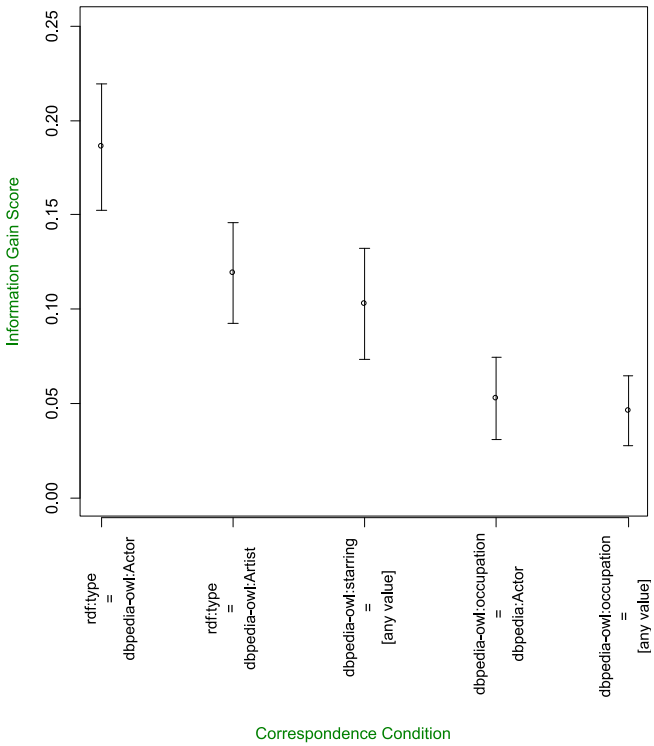

Fig 2. Top 5 ranked conditions for refining yago:Actor ⊑ dbpedia-owl:Person.

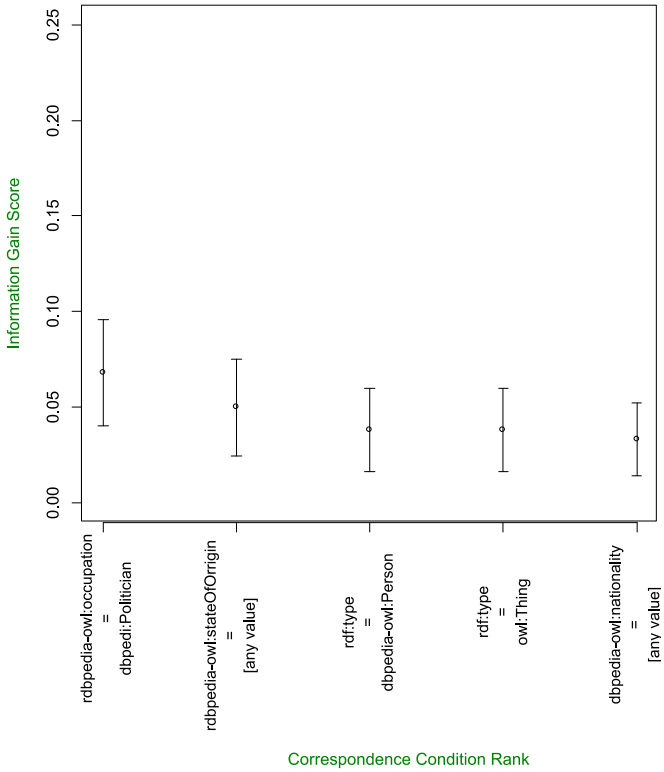

Fig. 3 Top 5 ranked conditions for refining yago:Politician ⊑ dbpedia-owl:Person

Mapping M_3: Our selection process performed poorly on this mapping, only returning the best correspondence pattern in first place 14% of the time. The pattern dbpedia-owl:occupation = dbpedia:Musician scored higher on average. While not our gold standard, on review this is could be seen as an acceptable pattern.

Mapping M_4: Our selection process performed also performed poorly on this mapping. As can be seen in figure 3, there was very little separating the mean IG scores for the top 5 ranked correspondences in this case. This case was difficult for the selection process because the attribute dbpedia-owl:occupation was set to dbpedia:Politician in relatively few instances, and there was in fact no attribute that could be used to reliably select politicians.

This demonstrates that for a relatively small training set of instance mappings, we can use the information gain measure to find good correspondence patterns. Even though the process does not always rank the best correspondence pattern the highest, it was shown to be capable of reliably ranking the best correspondence pattern within the top five.

VI. Conclusions

This paper proposes a method for discovering correspondence patterns that can be used to refine basic subsumption mappings between classes in independent ontologies. Our evaluation demonstrates that the Information Gain measure is a suitable scoring function for selecting correspondence patterns, provided a suitable training set of matched instances can be provided. This function allowed us to reliably select the best correspondence pattern as our top result in two of the four

mappings tested, and reliably returns the best correspondence pattern in the top five results for all four test cases. In one of the cases where the search algorithm did not return the best correspondence pattern as the top result (mapping M_3), this was because there were several patterns that could be considered valid and selecting the "best" among these was difficult. For the other case (mapping M_4) the attribute used in the best correspondence pattern was unreliably set in the instance data.

Our evaluation used training sets that were of a consistent size and quality. Further research will be required to address the question of how much training data and other background knowledge is required by our algorithm. Similarly, further evaluation will be required to establish how robust our approach is to errors in the training data such as misclassified instances. Our evaluation only considered selection correspondence patterns for four mappings, further evaluations will need to consider more mappings to allow us to more reliable recall and precision rates for our selection method.

In our evaluation, we limited our search to certain attributes, as we consider the full range of values for each attribute in our search. Before we can offer a general solution, we still need to address the question of how we can limit the range of values considered for each attribute, to make the search space tractable.

Our attribute selection method shows a promising start, but we do not see it as a stand-alone mapping solution. Our attribute selection process is intended to be used to refine mappings that have been discovered using existing semantic matching and mapping tools. Further work will be required to demonstrate its use in an overall mapping framework or process.

VII. Acknowledgement

This work is partially funded through the Science Foundation Ireland FAME Strategic Research Cluster (award No. 08/SRC/I1408), www.fame.ie.

References

[1] A. Ferrara, W. R. V. Hage, L. Hollink, A. Nikolov, and P. Shvaiko, "First results of the Ontology Alignment Evaluation Initiative 2011," *Informatica*, 2011.

[2] J. Hammer, M. Stonebraker, and O. Topsakal, "THALIA : Test Harness for the Assessment of Legacy Information Integration Approaches," in *Proceedings of the International Conference on Data Engineering (ICDE)*, 2005, no. August, pp. 485-486.

[3] W. Kim and J. Seo, "Classifying schematic and data heterogeneity in multidatabase systems," *Computer*, vol. 24, no. 12, pp. 12–18, 1991.

[4] F. Scharffe and D. Fensel, "Correspondence patterns for ontology alignment," *Knowledge Engineering: Practice and Patterns*, pp. 83–92, 2008.

[5] C. Bizer, "Linked data-the story so far," *International Journal on Semantic Web and Information Systems*, vol. 4, no. 2, pp. 1-22, Jan. 2009.

[6] C. Bizer et al., "DBpedia - A crystallization point for the Web of Data," *Web Semantics: Science, Services and Agents on the World Wide Web*, no. 7, pp. 154-165, Sep. 2009.

[7] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A large ontology from wikipedia and wordnet," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, no. 3, pp. 203–217, Sep. 2008.

[8] C. Fellbaum, Ed., *Wordnet: An electronic lexical database*. MIT Press, 1998.

[9] F. Giunchiglia, A. Autayeu, and J. Pane, "S-Match : an open source framework for matching lightweight ontologies," *Compute*, vol. 1, pp. 1-9, 2010.

[10] J. David, J. Euzenat, F. Scharffe, and C. Trojahn dos Santos, "The alignment api 4.0," *Semantic Web*, vol. 2, no. 1, pp. 3–10, 2011.

[11] D. Ritze, C. Meilicke, O. Sváb-Zamazal, and H. Stuckenschmidt, "A pattern-based ontology matching approach for detecting complex correspondences," in *Proc. of Int. Workshop on Ontology Matching (OM)*, 2009.

[12] H. Stuckenschmidt, L. Predoiu, and C. Meilicke, "Learning Complex Ontology Alignments A Challenge for ILP Research," in *Proceedings of the 18th International Conference on Inductive Logic Programming*, 2008.

[13] J. Lorey, Z. Abedjan, F. Naumann, and C. Böhm, "RDF Ontology (Re-) Engineering through Large-scale Data Mining," in *ISWC*, 2011.

[14] D. Hand, H. Mannila, and P. Smyth, *Data Mining*. Cambridge: MIT Press, 2001, p. 344.