Semi-automatic Integration of Knowledge Sources

Prasenjit Mitra
Department of Electrical Engineering
Stanford University
Stanford, CA, 94305, U.S.A.
mitra@db.stanford.edu

Gio Wiederhold, Jan Jannink
Computer Science Department
Stanford University
Stanford, CA, 94305, U.S.A.
{gio, jan}@db.stanford.edu

Abstract ***Integration of knowledge from multiple independent sources presents problems due to their semantic heterogeneity. Careful handling of semantics is important for reliable interaction with autonomous sources. This paper highlights some of the issues involved in automating the process of selective integration and details the techniques to deal with them. The approach taken is semi-automatic in nature focusing on identifying the articulation over two ontologies, i.e., the terms where linkage occurs among the sources. A semantic knowledge articulation tool (SKAT) based on simple lexical and structural matching works well in our experiments and semi-automatically detects the intersection of two web sources. An expert can initially provide both positive and negative matching rules on the basis of which the articulation is to be determined and then override the automatically generated articulation before it is finalized. The articulation may be stored or generated on demand and is used to answer customer queries efficiently.***

Keywords: information integration, knowledge discovery

1 Introduction

1.1 Need for an Ontology Algebra

Queries posed by end-users can, often, not be answered from a single knowledge source but require consulting multiple sources. When those sources are independent, the terms they use are not constrained to be mutually consistent. The semantic heterogeniety of these sources must be resolved in order to present a reliable and consistent view of the world.

The traditional approach to dealing with multiple knowledge sources is to create a unified schema or to merge the contents of these sources into an unified knowledge base [1], [2], [3]. Such an effort has two phases, first merging all the entries based on identical spelling and then manually resolving any recognized semantic mismatches. For instance, the erroneous match of *nail*, used in a human anatomy, with its use in carpentry must be undone. More complex are cases where definitions change over time, for instance types of cholesterol as disease-causing agents.

The merging approach of creating an unified source is not scalable and is costly. The process must be repeated when the sources change [4]. Furthermore, in certain cases a complete unification of a large number of widely disparate knowledge sources into one monolithic knowledge base is not feasible due to unresolvable inconsistencies between them that are irrelevant to the application. For a particular application, resolution of inconsistencies between a pair of knowledge sources is typically feasible, but it becomes nearly impossible when the objective is undefined and the number of sources is large.

Due to the complexity of achieving and maintaining global semantic integration, the merging approach is not scalable. We have adopted a distributed approach which allows the sources to be updated and maintained independent of each other. Articulations are generated between the sources that serve specific application objectives. Only the related articulations need to be updated when a term in the intersection of sources are changed and other articulations will remain as they were.

1.2 The Ontology Algebra

An *algebra* has been defined to enable interoperation between ontologies [5]. *Ontologies* are knowledge structures which explain the nature and essential properties and relations between terms present in a knowledge source. The *operators* in the algebra operate on selected portions of the ontologies. *Unary* operators like *filter* and *extract* work on a single ontology and are analogous to the *select* and *project* operations in relational algebra. They help us define the interesting areas of the ontology that we want to further explore. *Binary* operators that take as input two ontologies and outputs another ontology include *union*, *intersection* and *difference*. *Intersection* is the most crucial operation since it identifies the articulation over two ontologies.

1.3 Model of Articulation

In our model of articulation there are the sources, maintained autonomously by their experts, and applications, which need to use these sources. Linkages between sources are achieved through articulation contexts, which contain the terms that are needed to reach the sources and the rules which resolve semantic differences among them. The articulation contexts are created and maintained by articulation experts. For example, if an application has to access information from city maps, using local coordinates ranging from A1 to M19, and information from images, using latitude and longitude, the expert will provide the matching rules. When local maps are reissued, the coordinates will change if the city has grown, and these rules must be updated. Since the sources, say the local map, can be accessed remotely, we do not need to move all of the map detail into a knowledge base, as long as we can refer to its contents reliably. That means we do not need to update the coordinate mapping when the map is updated, say to indicate new buildings, but only when a major revision which changes its coordinates is made.

We find similar cases of interoperation requirements in purchasing of goods from multiple sources, in shipping, in genomics, etc. In all these instances we can also identify experts who must handle the interoperation of the diverse sources, although they have not been provided with specific tools for their task.

1.4 Organization

Section 2 discusses a tool (SKAT) that computes the articulation and introduces an example application based on government websites of NATO countries. Section 3 discusses the issues involved in computing the intersection and the techniques used in matching. Section 4 highlights the results obtained by matching two example NATO graphs. Section 5 points out the other uses of the matching tool. The last two sections conclude the paper and acknowledge other contributors to the work.

2 The Semantic Knowledge Articulation Tool

The articulation between two ontologies is determined using a semi-automatic *articulation tool*(SKAT). We use a subset of KIF [6], a simple first order logic notation to specify the declarative rules. The steps involved in computing the articulation are as follows:

1. The expert supplies SKAT with a few initial rules which indicate the terms that need to be matched and ones that must not be matched. For example, a rule

   ```
   (Match US.President FRG.Chancellor)
   ```

   would indicate that we want the US President to be matched with the German Chancellor. Similarly, a rule like

```
(Mismatch Human.nail Factory.nail)
```

would indicate that we do not want the term nail from the Human ontology match with the term nail in the Factory ontology. Apart from declarative rules, the expert has the option of supplying matching procedures that can be used to generate matches.

2. SKAT suggests matches and the articulation based on the supplied matching rules and the matching procedures approved of by the expert.

3. The articulation expert a) approves, b) rejects, or c) marks as irrelevant the suggested match or the rule used to compute the match.

4. SKAT now creates the correct rules and computes an updated articulation. The knowledge gained from the rejected or irrelevant rules and matches is stored by SKAT so as to avoid suggesting the same matches later.

2.1 An example: NATO websites

In order to demonstrate the concepts we have built a SKAT prototype that can be used by an application to identify the articulation between multiple web sources. Websites can be thought of as structured as a graph with a root page which has links to other related pages. Each website is structured differently and loosely represents an ontology. Initially the labelled graph structure of each website is constructed where each page is a node and all the links found on the page are modelled as outgoing arcs. A *label* is constructed for each arc from the title of the page that it points to and the anchor text found along with the link. We will illustrate our algorithm using examples that have been extracted from the government websites of NATO countries and show the matched nodes of the graph that constitute the articulation. (Figures 1, 2).

3 Intersection and Matching of Ontologies

The Intersection operator takes two ontologies and finds the correspondence of terms obtained from one ontology with that obtained from the other based on a set of rules. A major process in determining the intersection is to find this correspondence or *matching*. We will highlight the issues in matching and their solutions using object graphs obtained from the websites of NATO countries. It might be worthwhile to explore the types of mismatches that exist between ontologies that need merging. The typical mismatches that exist in such object graphs are as follows:

- *Term Semantic Mismatches*: these types of mismatches occur because two terms from two different ontologies refer to different concepts. Alternately, two different terms obtained from different ontologies might refer to the same concept. For the purposes of this work, we assume that within one particular ontology the same term consistently refers to the same concept.

- *Structural Mismatches*: these types of mismatches occur because the same term in one source matches multiple terms in another and causes one node in a graph match with many in the other. For example, the Prime Minister of a country might be holding the defence ministry too, whereas, in some other country the Prime Minister and the defence minister are different. In this case the node in the first graph will match against two nodes in the second. In order to allow for such mismatches between ontologies we allow a node in one graph to match multiple nodes in the second.

- *Instance Mismatches*: these mismatches occur because in an instance of a class in one source is not an instance of the same class in the second source. For example, one university considers its graduate students who hold assistantships as its employees, whereas another one does not. The US President is a part of the the government, however, the Finnish President is not. The Finnish Prime Minister is the head of the government and the President is a ceremonial head of state. The articulation rules should explicitly specify what matches we want to generate in such cases. In the absence of articulation rules, no matches will be generated.

- *Granularity Mismatches*: these occur when we have two matching nodes and the great grandchild of a node matches with the grandchild of the other node. This results because in the first graph a concept has been organized into a more elaborate hierarchy than in the second one. The expert can conservatively choose not generate any match between the intermediate nodes in the two graphs or decide to allow matching both the child and grandchild of the node in the first graph with the child node in the second one.

3.1 Expert Aided Matching

To generate the articulation the two ontologies need to be matched based on a set of *matching rules* specified by the expert. We do not automatically assume that terms from two different contexts match. This is necessary to avoid errors that occur due to the simplistic assumption that similarly spelled words have precisely matched meanings. If the expert believes that terms occurring in two different contexts are the same and relevant to the application, the expert can indicate that these terms mean the same across ontologies. The system will then generate the required matching rules. It is, however, our intent to generate an intersection that is minimal, i.e., it should contain only as much information as is necessary to compose knowledge from the two sources and answer queries posed by a particular application. We believe this approach will increase the precision of answers and reduce subsequent maintenance cost.

However, in cases where the ontologies are very closely related, most terms spelled the same might be referring to the same concept. In such a case, as an optimization, the expert might initially point out those terms spelled the same but are semantically different and then write a rule saying that whatever has not been indicated as mismatches till now are matches. In our example with NATO graphs we use the latter optimization.

3.2 Rule Based Semantic Mismatch Resolution

We envisage our higher-level rules to preprocess the terms in the labels and to determine the similarity of the terms. Such rules can declare global matching operations, as matching specified terms, requiring a dictionary or table lookup, or the use of a procedural function which matches terms. Such resources can be designed by the expert to satisfy recurring needs. However, rules that are sufficiently general in nature, especially procedural functions, can generate specific false matches which must be rejected or marked as irrelevant by the expert. If the rules are tuned towards the specific application contexts, fewer false matches will be generated.

3.2.1 Preprocessing Rules

SKAT initially attempts to match nodes in the two graphs based on their labels. Certain erroneous labels, especially, in the case of a single node having multiple labels, may be weeded out. In the graph for Canada, which has not been included due to space constraints, we had a node labelled 'Parliamentary System' twice and 'Senate' once and hence the former label was selected using a simple voting scheme. A better way is to analyze the document using IR techniques and determine what the referenced document contains. For most cases keeping both labels and matching with either to detect a correspondence does not generate too many false matches. Once again, the expert indicates whether we take the more conservative voting approach or the more generous approach of keeping all labels.

Another important preprocessing step is the removal of stopwords and stemming of words to their root words. The expert can choose and edit a list of stopwords and either provide a stemming procedure or specify a table (or individual rules) of root words.

SKAT uses rules specified by the expert to resolve semantic mismatches between two ontologies. In our example, the expert provided a matching procedure that simply matches terms from two ontologies if they are spelled similarly. However, before such a procedure is called, the expert had to take care of certain issues which otherwise would have produced false matches.

3.2.2 Context Identifier Tagging Rules

For the instance-level semantic mismatches as discussed above, we require to differentiate between the President of the US and that of Finland since, they are semantically different, yet the same term 'President' might have been used for both in the two graphs. This is achieved by a set of preprocessing rules, that simply indicate which terms need to be labelled with the context identifier tag. The labels of the nodes that pertain to the presidents of the two countries in the two graphs are tagged with the name of the contexts, i.e., they now become US.President and Finnish.President.

3.2.3 Context Identifier Removal Rules

The matching is performed based on a certain criteria. In our example graphs while matching between countries the expert might want to match the parliament nodes of two countries e.g., we want to match the node labelled 'Finnish Parliamentary System' with that labelled 'The UK Parliamentary System' in the graph pertaining to Finland. Therefore, a set of preprocessing rules can be supplied that reduce the labels such as 'Finnish Parliamentary System' to 'Parliamentary System' and thereby enable the matching.

3.2.4 Term Based Matching Rules

In our example an on-line dictionary or table specified by the expert can be searched to find matches among the terms. The terms that match in the two labels are given weights based upon their frequency of occurrence in the sources and other heuristics. A similarity score between two labels is calculated based on the match between terms in the labels. If the similarity score is above a threshold then the labels are matched.

Apart from the simple rules that simply mention the two terms that should be matched, the expert can supply more complex rules.

```
(Instance-Of Country UK)
(Instance-Of Country Finland)
(=> (and (Instance-Of Country ?Country1)
         (Instance-Of Country ?Country2))
    (Match ?Country1 ?Country2))
```

The first two sentences indicate that UK and Finland are countries and then provides a general purpose rule to match two countries. This has the added advantage that in order to match all countries to each other we do not have to list all combinations of matching rules. When we want to add another country all we need to do later on is to add the information that that country is an instance of Country.

3.2.5 Structure Based Matching Rules

Matching rules can also be based upon the structural similarity of the graphs. Parts of ontologies, represented by sub-graphs, can be matched based upon the similarity of their hierarchical structure. The matching rules are expressed as functions which take in the entire graphs and generate the correspondence between nodes of the two graphs.

Matching based solely upon structural similarity works well for sources that are structurally very similar. However, between most ontologies there is a fair degree of structural dissimilarity. Thus matching rules based only on the structure of the ontologies produce a variety of false matches. Therefore, the matching procedure first matches the nodes based on their labels and using this set of matches and the structural similarity, it generates further matches. In the strictest version, two nodes are matched if all their parent nodes and children nodes match and such a perfect match is given more weight.

The expert can specify the number of matches she is interested in generating and if the perfect structure match is not sufficient to identify the articulation then it is progressively relaxed. Nodes that do not match perfectly but have a low weight (i.e., a few of the parents and/or children match) are then accepted as being matched.

3.3 Other Matching Rules

In our example, instead of basing the matching only on the labels, SKAT can analyze the entire documents (i.e., web pages) and try to detect the similarity of the pages based on the words occurring in the page. A standard Information Retrieval algorithm can be used to generate such matches.

3.3.1 Spurious Match Resolution Rules

Techniques, such as automatic stemming, that were used previously can generate spurious matches. Words such as 'minister' and 'ministry' might have been preprocessed to be reduced to 'minister' and therefore, will match. However, for our government articulation we want to preserve the difference between the two. These sanity checking rules can be explicit statements of mismatches like

```
(Mismatch Minister Ministry)
```

provided by the expert that would indicate that we do not want certain matches. The expert can also indicate that certain phrases should not be preprocessed. Sanity checking rules like

```
(=> (and (Instance-of Person ?X)
         (Instance-of Office ?Y))
    (Mismatch ?X ?Y))
```

can also be used.

3.4 Performance Issues

Due to the more complex general purpose processing rules (the ones shown above that involve implication and logical inference), the performance of a system like SKAT can be really slow. Therefore, our prototype implementation of SKAT does not use those types of rules.

The structural matching procedures mentioned above are quadratic in complexity with respect to the number of nodes being matched. In case of very large graphs where such a complexity is unacceptable, structural slices of the graphs are matched against each other. It is expected that terms near the root of one ontology will match terms near the root of another and so on. Thus slicing the graphs should still generate an adequate articulation. The lexical matching is done by sorting all the terms in the labels in each graph and then matching is done in linear time. However, if sorting the terms in the entire set of labels is unacceptably slow for very large graphs, they too can be sliced and then matched.

4 Results

For the example graphs of Finland and UK the following is the match generated by SKAT using a simple lexical and term matching algorithm:

```
"Finnish Parliamentary System"
-> "The UK Parliament"
"Ministers"
-> "Lists of .. Ministers .."
"Government"
-> "Her Majesty's Government"
"Government"
-> "The Government"
"Ministers"
-> "Departments .. Ministers"
```

```
"Prime Minister"
-> "Prime Minister, ..  Service"
"Ministry of Defence"
-> "Defence (Ministry of"
"Parliament"
-> "The UK Parliament"
```

where the first label refers to a node in the Finland graph and the second to that in the UK graph. As we can see, most matches are correct and intuitive. Using the structural matching algorithm we are able to generate the matches between

```
"Ministries"
-> "Departments"
"Ministries"
-> "Desc. of Ministry of Defense"
```

This was after the requirement for perfect structural match was relaxed and granularity mismatches were not resolved conservatively. Since the Government sites and the Department of Defence sites in both the graphs matches, the nodes in the paths between these two nodes i.e. 'Ministries' and 'Departments' were matched. The unwanted match generated 'Ministries' with 'Desc. of Ministry of Defence' is the price we pay for relaxing the perfect match requirement. A more conservative approach would generate no structural match.

5 Other Uses of the Matching Tool

For our examples, the generated articulation consisted of extracted structural information from the government sites of the NATO countries, and the important nodes selected from them. The result is a partial skeleton of the entire website, providing a taxonomy of the site, as well as of governmental structure.

The government websites were laid out in a hierarchical fashion and the portions of the hierarchy that were common across sites were extracted out. The common portion of the second graph can be restructured so that the nodes are arranged according to the hierarchical structure of the common portion of the first graph. This restructuring, creates a view of the second graph on the lines of the first and can be stored. It is now easier to answer user queries that require searching multiple websites since we have reformatted the articulation of the sites into one common structure.

SKAT can be used to extract information from a website by supplying a template graph whose nodes are labelled with terms of interest. For example, a graph constructed from one of the existing government ontologies can be used as a template graph and its articulation with the Finland graph will give us the nodes in the Finland graph that correspond to the terms in the template graph. A simple adaptation of SKAT can thus be used as a template-based querying tool wherein the answer will be structured according to the provided template.

Since SKAT extracts structural information from an ontology it can be used to create a new ontology. If a graph has little structure we can compute the articulation of this graph with an already existing structured graph. Using the articulation we can structure portions of the first graph. Given the huge amount of information present in today's World Wide Web, one can just supply SKAT the root node of a country's graph or a reference ontology and a set of webpages and those pages will be automatically structured.

Once web pages from distinct sources are consistently structured, queries by end-users will be reliably answered. Misleading matches will be avoided and many new matches, that are now based on verified semantic identities will be created.

6 Conclusion

Applications and their decision-makers benefit from broad access to information, but the information is widely dispersed and difficult to integrate reliably.

We are addressing an important problem in the use of the many diverse knowledge sources that are available to our applications.

By keeping the intersection as small as feasible we reduce the maintenance costs for the applications and maximize the autonomy of the sources. By allowing sources to remain autonomous we can take advantage of the maintenance efforts made by independent, specialized experts.

Tools, as SKAT, to create modest and managable articulations of these sources for well-understood applications allow application experts to maintain the linkages needed for reliable interoperation. Such reliability is a requirement for business-transactions, since we cannot expect human filtering and matching to occur with regular, repeated operations. The investment made once, by the articulation expert, will pay off every time disjoint domains are used to process an order or make a business decision.

7 Acknowledgements

Thanks are due to P. Srinivasan for their help in preparing the draft of the paper and comments and suggestions to improve it.

References

[1] C.A. Knoblock, S. Minton, J.L. Ambite, N. Ashish, P.J. Modi, Ion Muslea, A.G. Philpot, and S. Tejada. Modeling web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence, Madison, WI*, 1998.

[2] M.R. Genesereth, A.M. Keller, and O.M. Duschka. Infomaster: An information integration system. In *ACM SIGMOD '97*, pages 539–542. ACM, 1997.

[3] A.T. McCray, A.M. Razi, A.K. Bangalore, A.C. Browne, and P.Z. Stavri. The umls knowledge source server: A versatile internet-based research tool. In *Proc. AMIA Fall Symp*, pages 164–168, 1996.

[4] D.E. Oliver, Y. Shahar, E.H. Shortliffe, and M.A. Musen. Representation of change on controlled medical terminologies. In *Proc. AMIA Conference*, Oct. 1998.

[5] Gio Wiederhold. An algebra for ontology composition. In *Proceedings of 1994 Monterey Workshop on Formal Methods*, pages 56–61. U.S. Naval Postgraduate School, September 1994.

[6] M.R. Genesereth and R.E. Fikes. *Knowledge Interchange Format,*. Reference Manual, 1992.

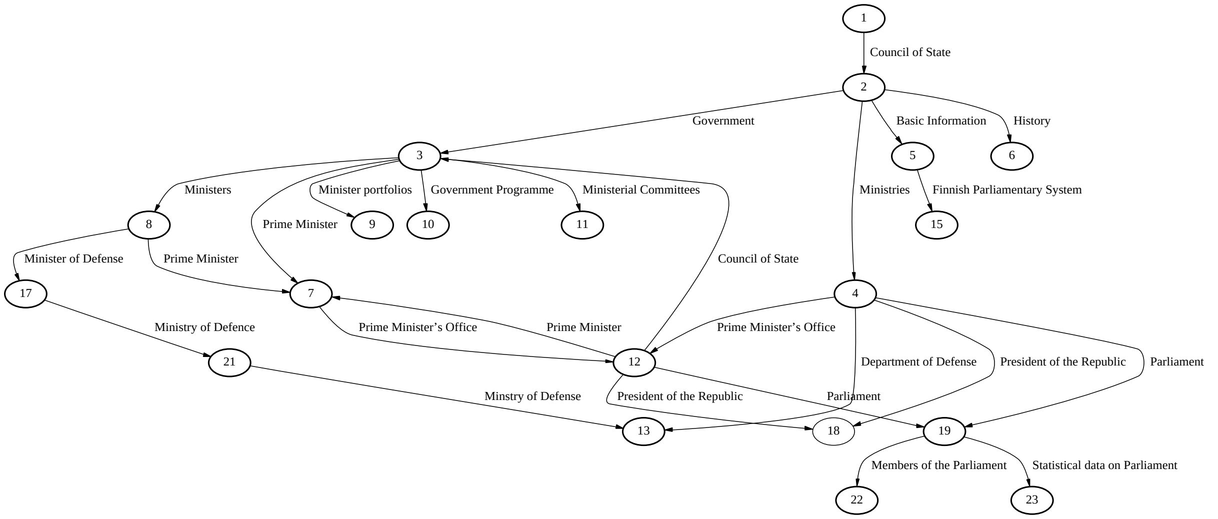

Figure 1: Partial Graph of Finland

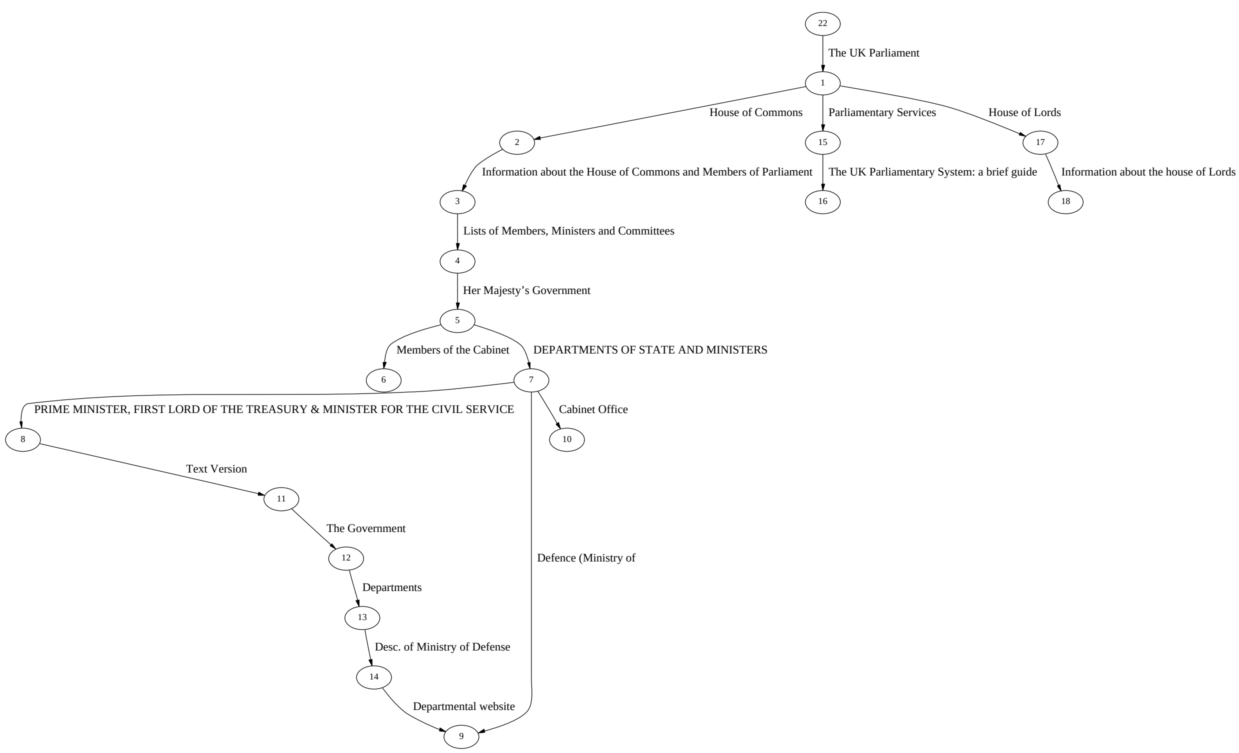

Figure 2: Partial Graph of UK