Federated Ontology Search

Vasco Calais Pedro, Eric Nyberg, Jaime Carbonell
Language Technologies Institute
School of Computer Science
Carnegie Mellon University

Abstract

In this paper we describe a novel methodology for retrieving and combining information from multiple ontologies. In the last decades the number and diversity of available ontologies has grown considerably. There may be a variety of such resources available, but the cost to integrate them into an application is incremental, and often prohibitive for exploratory prototyping and discouraging for larger-scale integration. However, a unified cross-ontology query interface provides the capability for searching them independently and combining results. Beyond a common interface we further explore combining ontological graph search across knowledge bases and demonstrate superior results in the context of type-checking for automated Q&A systems. The paper presents the method and our initial promising results.

1 Introduction

In the last decades the number of available ontologies has grown considerably. Several proprietary and open-domain efforts such as Cyc [1], SUMO [2], Omega [3], Scone [4], ThoughtTreasure [5], have become available. . Swoogle [6] has now indexed more than 10 000 ontologies. These resources offer the promise of easily-accessible, open-domain ontological information, but the existence of such diverse ontologies raises the issue of information merging and reuse. A comparison of the ontologies reveals both redundant and complementary coverage, but the variety of frameworks and languages used for ontology development makes it a challenge to merge query results from different ontologies. The number of available languages for ontological knowledge engineering such as RDF, OWL, DAML+OIL and CYCL, combined with the existence of independent interfaces aggravates the issue. The lack of a formal way to access and combine the knowledge from different ontologies is an obstacle to more effective re-use and combination of these resources.

One approach to the multi-ontology issue is to absorb all the knowledge into a common ontology ahead of time. However this approach as several drawbacks, as defined in [7; 8], such as (i) non-scabality, (ii) losing language and reasoning specificity of distinct ontologies, (iii) losing privacy and autonomy of ontological knowledge (iv) language level mismatches such as syntax mismatches, differences in logical representation and different semantic primitives and (v) Ontology level mismatches, such as difference in scope, coverage and granularity, making this challenge thus far too daunting in practice. Another approach is to punt on the merging problem and simply query more than one ontology via different interfaces, and interpret the results of each ontology individually, essentially moving the entire challenge from the ontology provider to the application builder. A third approach is to build an ontological middleware level for only small fragments of ontologies in an on-demand basis, that is:

- Query multiple ontologies and then merge the query results from multiple knowledge base systems, much like Federated Search in information retrieval [9].
- Follow ontological chains and inferences across ontologies, using partial query results from one ontology to query another. This is a more complex version of cross-data-base joins, where the data schemas are sufficiently compatible.

Currently, the main approaches to a solution for these problems focus on ontology integration, by creating a mapping between the concepts and relations of different ontologies. In some cases, such as the Semantic Web project [10] primarily rely on merging two ontologies by establishing a full mapping between them. Some efforts have tried to produce a merged ontology automatically using a bottom-up approach such as FCA-Merge [11]; most involve some degree of semi-supervised mapping. Other approaches, such as the one taken by CYC, try to absorb other ontologies into a single main ontology while maintaining coherence [12]. One disadvantage of these approaches is the prohibitive cost of producing a mapping or absorbing an ontology, given their increasing scale and rate of availability. Another disadvantage is that it's not always possible to establish a one-to-one mapping between the concepts and relations in one ontology and the concepts and relations in another. Furthermore, there is the problem of keeping the mappings updated as the original ontologies evolve. A large number of available ontologies are considered works in progress and are updated frequently, which implies a constant updating of any mappings associated with those resources.

Most applications that use ontological information would benefit from an approach that models the information need, queries the relevant ontologies and retrieves the best result while providing a single unified interface to the client application. If we look to other domains for inspiration on how to proceed, we can find a similar problem in the field of Federated Search [13; 14]. Information Retrieval is usually based on a single database model of text retrieval. But to cope with proprietary information spread around the world in separate databases, distributed information retrieval explicitly models multiple databases for text retrieval. Each database is queried independently, the results are merged when possible and a new global ranking is established. In the same fashion, we can model our ontologies as individual sources, construct a query that describes the information need, query each ontology independently and merge the results into one ranked list.

Using Federated Ontology Search we can parallelize query execution while respecting the structure of the individual ontologies, taking advantage of both redundant and complementary knowledge in the available ontologies to improve the overall performance of the system.

2 Ontological Search

The success of the proposed approach hinges on the definition of a search method that is independent of any ontology. For this purpose we introduce the concept of *operator* and a concept of query based on operators. The main purpose of an operator is to decouple the search process from the information need. Instead of describing a complete semantic framework, the goal is to describe the information request in terms of a decomposable query that can be transformed into a set of operators. This would provide an elegant abstraction from the formal representations implemented by our ontological sources, allowing each operator to be an independent request.

As an example, Figure 1 shows the execution of the query *children(car)*. This query uses the *children* operator to get all the children of the concept *car*.

It is important to note that by defining a set of operators we are in fact delegating responsibility for their execution to the ontologies themselves, therefore making no restrictions on whatever processes are executed in order to obtain the necessary information. This means that operators can be implemented using extended features of ontologies (e.g. inference, grounding, restrictions and theorem-provers). The only constraint is that the output of each query execution is a Rooted Directed Acyclic Graph (RDAG).

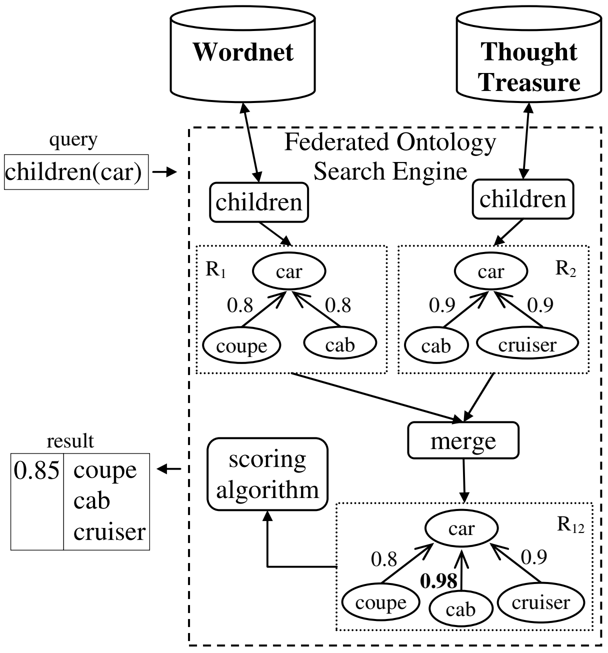

Figure 1 - Operators in Federated Ontology Search

2.1 Operators

We proceed to describe the set of operators that were used for our particular evaluation, as given by this simplified description for purposes of clarity;

- **rel(a,b,rels)** The relation operator returns a graph containing a chain linking *a* and *b* using the relations contained in rels.
- **parents(a)** Returns the graph containing the parents of *a*.
- **children(a)** Returns the graph containing the children of *a*.

All operators must have the property of compositionality in order to enable the constructions of complex operations based on simpler ones. The ideal operator set should contain the operations usually performed in ontologies. Providing the full set of operators is not the scope of this paper but rather future work.

There are three sub-problems described in the literature on Federated Search that also apply to this area: *resource description*, *resource selection* and *results merging and scoring*. In the ontological domain, resources are ontologies.

3 Ontology Description and Selection

The goal is to select a subset of the available ontologies based on some measure of appropriateness of an ontology given a query. This can be modeled as the probability p(O,q), where O is an ontology and q is a query.

Distributed online efforts such as OpenMind result in ontologies that are constantly updated thus making the task of modeling them very difficult. We must also consider cases where the ontologies make use of inference engines and logic mechanisms.

Therefore, a more general approach is to evaluate the relative utility of different ontologies by comparing the results they produce for a given input query. Although theoretically we could create a gold set of queries to perform on each ontology and use the results to create an ontology model, due to the diverse nature of ontologies this is in practice a very hard task. At this point we use a naïve approach where the query is executed in the subset of ontologies that contain the concepts in the query, but this can bring problems in both performance and accuracy. In order to mitigate this risk, we created a parameter that models the general accuracy for a given source. This parameter is a good candidate for the application of a learning algorithm such as some variation of the Expectation Maximization algorithm [15].We need to create gold standards to train and test such algorithms, which makes the process time consuming.

4 Merging and Scoring

One advantage of this approach is that rather than trying to merge two entire ontologies, we merge only the results. This significantly reduces the problem of merging ambiguous concepts given that queries are grounded in a query concept and therefore only relations that apply to the concept in the query are returned. Polysemic concepts will be explicitly modeled via separate results, with a variable confidence on the relations contained in each result.

The structured nature of the query results implies that the result merging problem in the ontology domain is very different from result merging in federated text search, where the results typically consist of unstructured data. In the case of Ontology Search the results are graphs, which allow us to treat merging as an instance of Inexact Graph Merging. While one of the goals of merging in Federated Search is to eliminate duplicates, the primary goal of merging in ontology search is to find complementary information. The goal of ranking in ontology search is to produce, at the top-ranked position, the most complete and accurate result. Generally speaking, we want to merge two results if they represent information about the same concepts, thus creating a more complete result.
As an example, let us consider three distinct ontologies, O_1, O_2 and O_3. Let us assume that we execute the query *sim(bank)*, with the purposes of finding concepts similar to bank. Imagine that ontology O_1 and O_2 both interpret *bank* as *river_bank* while ontology O_3 interprets *bank* as *money_bank*. Given this scenario, we would like the result set to contain two results, one referring to the similar concepts of *river_bank* and the other to the similar concepts of *money_bank*. The results from O1 and O2 should be

merged since that is likely to yield an increase in the quantity of information contained in the result, as well as the confidence in the concepts common to the two results.

The problem can be formulated as follows. Given two RDAGs g_1 and g_2 we want to merge the two graphs if they are similar. Typically this is done by considering one of two options, either we measure the similarity between the graphs or measure the difference. But in our case we not only want to measure the similarity but also find the maximum common subgraph. We should look to the science of Inexact Graph Matching for guidance here.

Inexact Graph Matching occurs when we do not expect to find an isomorphism between the two graphs to be merged. This is one of the most complex problems in computer vision [16] and is also an important part of chemical similarity searching [17]. More specifically, inexact graph matching is proven to be an NP-Complete problem [18].

In the next section we will discuss graph similarity.

4.1 Graph Similarity

Graph similarity Distance [19] is typically calculated in one of the following ways: Cost Based Distance, Feature Based Distance or Maximum Common Subgraph.

Cost Based Distance is based on edit operations on the graph, typically add nodes or edges, remove nodes or edges and re-label nodes or edges, where each operation is associated with a cost. Given two graphs g_1 and g_2, the edit distance between g_1 and g_2 is the minimum number of edit operations necessary to transform g_1 into g_2.

Feature based distances use a set of invariants established from the graph structural description, using these features in a vector representation to which we then apply distance or similarity measures.

The goal of the Maximum Common Subgraph approach is to find the largest Subgraph common to both g_1 and g_2. To address this requirement, current approaches use the concept of maximum clique detection. The problem is then changed into finding the Maximum Common Edge Subgraph. In our case we use a variation of the overlapping coefficient for graphs, a measure whereby if graph g contains g' or the converse then the similarity coefficient is a full match.

4.1.1 Localized Confidence Boosting Algorithm

Given g1 and g2 as results of a query, the algorithm is as follows. After applying the screening procedure we are left with graphs where $sim(g_1,g_2) > T$. Given that we now basically have $g_1 \cap g_2$ we will apply localized boosting and then add the nodes and edges that were previously discarded.

The basic intuition behind the confidence boosting is that the confidence of the edges is boosted whenever two edges are merged. The boosting is determined through the use of the *Soft Or*, given by the formula:

$$1 - \prod_i (1 - c_i)$$

In order to apply confidence boosting we apply the concept of tuples, where $t_x = (c_x,c_y,r)$ is a tuple, c_x,c_y are concepts and r is a relation. First we split g_1 and g_2 into tuples $t_x = (c_x,c_y,r)$, $c_x,c_y,r \in g$, such that c_x and c_r are adjacent and $r(c_x,c_y)$. We then compare the sets of tuples from g_1 and g_2 and if $sim(t_x,t_y)>T$ then we boost the confidence of t_x.

4.1.2 Tuple Similarity

Tuple similarity measures are based on the linear combination of the edge similarity and the concept similarity.

When comparing concepts or relations, we use the Q-Gram distance [20] on the strings that represent them. The intuition behind the use of q-grams as a foundation for distance metric is that when two strings s_1 and s_2 are within a small edit distance of each other, they share a large number of q-grams in common. This metric is fairly robust to orthographic errors, morphological errors and compound words, which makes it suitable for our purposes. The similarity between two tuples is given by the minimum similarity of the concepts and relations contained in the tuples. Formally

$$sim(t_x,t_y) = \min\begin{cases} sim(c_{x1},c_{y1}) \\ sim(c_{x2},c_{y2}) \\ sim(r_x,r_y) \end{cases}$$

4.2 Scoring Results

A result is scored in a compositional manner, by scoring the outcome of each operator used in a query individually before calculating the final score. A ranking will be computed from the scores of the results thus making the computation of the ranking score a key issue, much like in traditional Information Retrieval.

Each operator focuses on either recall or precision. Operators that focus on recall will typically return results with as much information as possible. The *similarity* operator, for example, returns all the synonyms associated with a concept, the more synonyms the better the result should be, all else being equal. Operators that focus on precision will usually return chains of associations. They focus on precision of the relations. As an example, the *relation* operator finds the relation between two concepts. All else being equal, a direct relation would be better than a long chain of relations.

When a result is merged from two other results, the confidence in the sources from were the results where extracted is combined using the *soft or* rule, as described before. We now present two scoring metrics, to be used by precision type operators and recall type operators respectively.

Precision scoring metric The goal of this scoring metric is to give preference to shorter graph lengths. Given that this metric is used for precision type operators, it is desirable to have results with short chains.

$$S(r) = C_s \frac{\prod_i (C_{ei})}{\left((avg\_lenght)^2\right)^2}$$

Where c_s is the confidence of the source, c_e is the confidence on the edge and *avg_length* is the average distance of the paths contained in the graph, from root to leaf node.

Recall scoring metric The recall scoring metric gives preference to graphs with large node degrees. Given that this metric is used by recall type operators, we want the score to increase with the quantity of diverse information contained in the graph. Thus

$$S(r) = C_s \times \left(1 - \frac{1}{\left((avg\_\deg ree)^2 \right)^2} \right) \times \prod_i (c_{ei})$$

Where avg_degree is the average degree of the nodes contained in the result.

5 Preliminary Results

In this section we present some retrieval experiments using the federated approach to ontologies. One possible evaluation of the proposed approach requires a task centered evaluation process. For our experiments we selected the task of type checking, described below.

5.1 Type Checking

Type checking using federated ontology search can be viewed as the task of finding an *is-a* based path between two concepts. Our approach has the advantage of using indirect paths when no direct path is found. An indirect path consists of partial ordered sub paths that exist in separate ontologies but form one path when combined. Finding an indirect path is possible by simply applying either the *parents* operator or the *children* operator to the source node in one ontology and using the resulting nodes to query for a direct path in another ontology. The resulting path is the combination of these partial paths. Using indirect paths provides a promising way of combining information that by itself would be incomplete and enabling the deduction of previously inexistent paths.

5.2 Experimental Setup

A total of 9558 pairs were extracted from results of the Javelin question answering system in TREC QA 2003 [21]. Each pair consists of the expected answer type or subtype and the candidate answer. For the purposes of our evaluation we used two of the currently available ontologies, Wordnet and ThoughtTreasure. The purpose of this preliminary evaluation is to contrast the performance of each of the ontologies individually, which would be a typical scenario for a project using one ontology as a knowledge base, with the performance of the set of ontologies using a federated approach.We have evaluated the recall and precision of the retrieved results.

5.3 Results and Analysis

Figure **1** shows the recall after running the test set with different configurations.

Configuration	Recall
Wordnet	4278 (44.7%)
ThoughtTreasure	730 (7.6%)
Combined	4686 (49%)
Merge	4686 (49%)

Merging + Indirect	6870 (71.8%)
Test size	9558

Table 1: Recall using different configurations with the full set of pairs

Wordnet and *ThoughtTreasure* were experiments where Wordnet and ThoughtTreasure were used individually. The *Combined* experiment queried each of the ontologies individually, picking only the top ranked result. The recall is lower than the direct sum of the individual results due to knowledge overlap in the ontologies. The *Merge* experiment queries both ontologies but merges the results using the merging algorithm described previously. Finally we use merging as well as indirect path query to perform the last experiment Although the recall remained the same when applying the merging procedure, the average confidence of the top result, in cases where there was more than one result, increased significantly (28%), as shown in Table 2.

	avg. confidence
Without merging	0.72
With merging	0.93
increase	28.7%

Table 2 – The Increase in the average confidence of the top ranked result due to the merging algorithm.

In order to test the accuracy of the federated approach, we created a gold standard for a subset of the full set of pairs. Using random sampling, we selected 1300 pairs, which we then proceeded to judge them manually. For each pair in the gold standard subset we generated a tuple of the form *(type, concept, judgment)*, where judgment reflects if the *concept* is of the type *type*. We compared the answers of the Federated Search with the gold standard by applying a variation on the result score threshold. If a score is below the threshold then the *concept* is considered not to be of the type *type*. We obtained a significant increase in performance when using the federated search approach. The optimal threshold for this experiment is $T=0.1$ with a precision of $P = 0.676$. The recall was very close to the one obtained in the full set with a recall of 0.71 (71%). Below we can see the F-Measure of the system.

Precision and Recall gold standard set	
Precision	0.71
Recall	0.67
F1 Measure	0.69
Baseline F1 measure	0.53
Increase	30.18%

Table 3 – Precision and recall of the Federated System using Wordnet and ThoughtTreasure

6 Related Work

Many approaches deal with the same issues as our work. The FCA-Merge algorithm [11] relies on the existence of a common corpus and is not a fully automatic procedure. The IF-Map method [22], the PROMPT system [23] have the goal of creating an alignment between ontologies, thus facing the drawbacks mentioned before. SWOOGLE [6] allows for the retrieval of ontologies, but it does not merge the information contained in them. The most similar work to our knowledge is DRAGO [7], which uses the peer-to-peer paradigm with Distributed Description Logics to supply distributed reasoning services in multiple ontologies. This approach, although very promising, does not permit a centralized merging procedure stemming from one centralized query. This makes it hard to use for tasks that rely on the knowledge contained in the ontologies.

7 Challenges and Future Work

Concept and edge similarity is currently calculated via string matching. This assumes that the different ontologies are using a similar namespace, were every ontology will have similar names for similar concepts. The current method is not robust when the relations used in different ontologies are extremely different or with ontologies using different languages. Although this is currently an open ended problem, a possible solution is to compare the structures in which the two concepts occur to determine similarity as a back off strategy. Ontology description in dynamic ontologies also presents a challenge. In the future we expect to model and ontology based on the use of random queries to determine the domain area of a given ontology.

8 Conclusions

In this paper we describe a novel methodology for retrieving and combining information from diverse ontologies. This methodology presents several benefits over full merging procedures such as the FCA-Merge method by reducing the potential for ambiguity by merging only local results which are grounded in query terms. This method addresses the issue of dynamic ontologies, in which the continuous update to the ontology makes it costly to adopt a merge approach. We establish a parallel to federated search in information retrieval and proceed to describe the algorithms for merging and scoring the results from multiple ontologies and we introduce a preliminary set of operators, currently implemented in a set of ontologies. We perform an evaluation for federated ontology search within the task of type checking and show that this method outperforms individual ontologies both in precision and recall. Although there are many challenges to be solved, this is a promising approach that brings closer the goal of easier integration of information from different ontological sources.

9 Acknowledgements

We would like to thank Maria Calais Pedro for the creation of the gold standard test set, Andy Schlaikjer for the multi-client java server code contributed to the system implementation and Jeongwoo Ko for the help in evaluating the system. This work was supported in part by the Fundação para a Ciência e Tecnologia, grant SFRH / BD / 5171 / 2001.

References

[1] D. B. Lenat, "CYC: a large-scale investment in knowledge infrastructure," *Communications of the ACM,* vol. 38, pp. 33-38, 1995.

[2] I. Niles and A. Pease, "Towards a standard upper ontology," *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001,* pp. 2-9, 2001.

[3] A. Philpot, M. Fleischman, and E. H. Hovy, "Semi-Automatic Construction of a General Purpose Ontology," *Proceedings of the International Lisp Conference. New York, NY. Invited,* 2003.

[4] S. E. Fahlman, "Scone user's manual," 2005.

[5] E. T. Mueller, "Natural Language Processing with Thoughttreasure," Signiform, 1997.

[6] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: a search and metadata engine for the semantic web," *Proceedings of the Thirteenth ACM conference on Information and knowledge management,* pp. 652-659, 2004.

[7] L. Serafini and A. Tamilin, "Drago: Distributed reasoning architecture for the semantic web," *Proc. of the Second European Semantic Web Conference (ESWC'05),* vol. 3532, pp. 361–376, 2005.

[8] M. Klein, "Combining and relating ontologies: an analysis of problems and solutions," *Workshop on Ontologies and Information Sharing, IJCAI,* vol. 1, 2001.

[9] L. Si and J. Callan, "Modeling search engine effectiveness for federated search," *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval,* pp. 83-90, 2005.

[10] T. Bemers-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American,* vol. 284, pp. 34-43, 2001.

[11] G. Stumme and A. Maedche, "FCA-Merge: Bottom-up merging of ontologies," *7th Intl. Conf. on Artificial Intelligence (IJCAI'01),* pp. 225–230, 2001.

[12] S. Reed and D. Lenat, "Mapping ontologies into cyc," *AAAI 2002 Conference Workshop on Ontologies For The Semantic Web, Edmonton, Canada, July,* 2002.

[13] J. Callan, "Distributed information retrieval. Advances in information retrieval," Kluwer Academic Publishers, 2000.

[14] D. Fryer, "Federated search engines," *Online(Weston, CT),* vol. 28, pp. 16-19, 2004.

[15] T. K. Moon, "The expectation-maximization algorithm," *Signal Processing Magazine, IEEE,* vol. 13, pp. 47-60, 1996.

[16] E. Bienenstock and C. von der Malsburg, "A neural network for invariant pattern recognition," *Europhysics Letters,* vol. 4, p. 121, 1987.

[17] J. W. Raymond, E. J. Gardiner, and P. Willett, "RASCAL: Calculation of Graph Similarity using Maximum Common Edge Subgraphs," *The Computer Journal,* vol. 45, pp. 631-644, 2002.

[18] A. M. Abdulkader, "Parallel Algorithms for Labelled Graph Matching," PhD thesis, Colorado School of Mines, 1998. Understanding and Analysis–MIUA 2001, 1998.

[19] A. Sanfeliu and K. S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 13, pp. 353-362, 1983.

[20] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, L. Pietarinen, and D. Srivastava, "Using q-grams in a DBMS for Approximate String Processing," *IEEE Data Engineering Bulletin,* vol. 24, pp. 28-34, 2001.

[21] E. Nyberg, T. Mitamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-Thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, and S. Judy, "The javelin question-answering system at trec 2003: A multi-strategy approach with dynamic planning," *Proceedings of the Twelfth Text REtrieval Conference (TREC2003),* 2003.

[22] J. Barwise and J. Seligman, *Information flow: the logic of distributed systems*: Cambridge University Press, 1997.

[23] N. F. Noy and M. A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," *Proceedings of the National Conference on Artificial Intelligence (AAAI),* 2000.