From: FLAIRS-02 Proceedings. Copyright © 2002, AAAI (www.aaai.org). All rights reserved.

An Approach for Semantic Search by Matching RDF Graphs[1]

Haiping Zhu, Jiwei Zhong, Jianming Li and Yong Yu

Department of Computer Science and Engineering
Shanghai JiaoTong University
Shanghai, 200030, P.R.China

{zhp036, zjw035, ljm038}@mail1.sjtu.edu.cn, yyu@mail.sjtu.edu.cn

Abstract

The World Wide Web is developing rapidly, but neither recall nor precision of traditional search engines can satisfy the increasing demands of users. Presently, RDF is widely accepted as a standard for semantic representation of information on the Web, which makes possible the advanced search among web resources. In this paper, we introduce an approach for semantic search by matching RDF graphs. New similarity between RDF graphs is defined and ontologies on arcs as well as on nodes are employed. The implementation of a demonstration system on our method is currently in progress.

1 Introduction

With the exponential growth of the Web, information retrieval and resource discovery is getting more and more challenging. However, traditional search engines, the majority of which are based on keyword matching techniques, have inherent defects. They seem more competent to perform full-text analysis and search for user-specified keywords, but fail to exploit and consequently retrieve the content of web documents. As a result, neither recall nor precision can satisfy the increasing demands of users despite their persistent efforts on technique development. For better performance, we need radically more intelligent search techniques.

Recently, semantic search has become a research hotspot. As shown in OntoSeek (Guarino, Masolo, and Vetere 1999), the combined use of linguistic ontologies and structured semantic matching can improve markedly both recall and precision. Designing such a system, we always need to first extract semantic information from online documents to make it understandable by machines ahead of performing semantic matching. Since RDF (Lassila and Swick 1999) has been widely accepted as a standard for semantic representation for the next generation of the Web, we propose our semantic search approach based on matching RDF graphs. In our method, first we collect information of a certain domain from the Web and build up our resource RDF graph repository. Afterwards, when the user enquiry sentence is entered, it is interpreted into a query RDF graph. Then we come to the key step of computing the similarity between the query graph and each candidate resource graph. The similarity definition is based upon the ontology which consists of type hierarchies on nodes and arcs. Finally, the matching results are ranked and orderly returned to the user.

The rest of the paper is organized as follows. Section 2 introduces some basic concepts that ground our research on semantic matching. Section 3 formally defines the semantic similarity and describes in detail our graph matching algorithm with a running example. Section 4 gives evaluation and discussions on our algorithm. Section 5 concludes our approach through a comparison with some related work.

2 Basic Concepts

2.1 Semantic Representation

Before semantic search can be fulfilled, semantic structures from unstructured information sources should be acquired first. We know that most web documents are described in natural language and users also prefer to enter natural language queries to retrieve them. Therefore, the ultimate task amounts to generate structured semantic representation from natural language sentences.

To achieve automatic generation, we restrict the field into a specific domain, say clothes descriptions, so that we can benefit from distinct domain characteristics. Zhang (Zhang and Yu 2001) has proposed a machine-learning based approach that can be trained for different domains and requires almost no manual rules. In his method, Conceptual Graph (Sowa 1984) serves as the tool for semantic representation while WordNet (Miller 1990) version 1.6 together with the manually constructed relation hierarchy acts as the domain ontology in the whole process of semantic analysis and extraction. Since some other researches have shown that there exist direct mapping and

Copyright © 2002, American Association for Artificial Intelligence (http://www.aaai.org). All rights reserved.

[1] This work is supported by IBM China Research Lab.

lossless conversion between Conceptual Graph and RDF graph (Berners-Lee 2001; Corby, Dieng, and Hébert 2000), Zhang's method is considered transferable to the generation of RDF graphs. A prototype named ALPHA has been developed, and original results from it demonstrated the feasibility of the approach (Li, Zhang, and Yu 2001). According to the transferability, we will use "concept/relation" and "node/arc" interchangeably in the remainder of the paper.

2.2 Entry of Graph

General matching will easily suffer from the NP-complete problem of *Maximum Subgraph Matching*, so including some user input to guide the matching process is reasonable and important. *Entry of graph* embodies our above thought in the design of our system.

Entry of Graph. With ALPHA, we have comprehended clothes descriptions from online shops into RDF graphs. In each resource graph, the node representing the corresponding clothes category, e.g. *shirt*, *dress*, *pants*, etc., will be designated as the *entry of graph*. Namely, the entry node will serve as the start position during further graph matching process.

Similarly, each query graph will also have its entry. Our system requires users to give the entries in their queries. Though some existent techniques, such as "shallow parsing", can help detect the implied "headword", the extra work on tailored rules or exhaustive training lead us back to direct interaction with users at the initial stages of our work.

Ontology Based RDF Graph Index. Resource RDF graphs will be indexed in WordNet, the domain ontology, according to their entries. The index naturally confines matching range to separate clothes categories, for graphs have been clustered by their entries.

Besides, we also use the index to enable hierarchical matching.Since hyponymy and hypernymy of senses in WordNet can be regarded as subsumptive relationship between graph entries, the index makes it possible to search all categories subsumed by the user's querying object.

jersey, T-shirt, tee shirt
=> **shirt**
=> garment
=> clothing, clothes, apparel, vesture, wearing apparel, wear

Fig. 1. An ontology segment in WordNet.

For instance, figure 1 shows an ontology segment in WordNet. When a user inquires about a "shirt", not only resource RDF graphs about "shirt" but also the ones about "jersey" or "tee shirt" will be searched to match the query, although they are quite different in word forms.

3 Semantic Search by Matching RDF Graphs

In this section, we will present our approach that performs the semantic search by matching RDF graphs. Let us take the matching task of the following query graph (figure 2) and resource graph (figure 3) as an example.

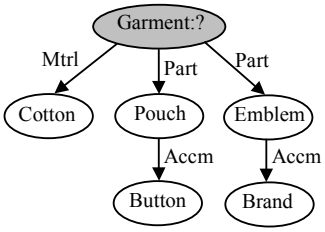

Fig. 2. Query RDF graph.

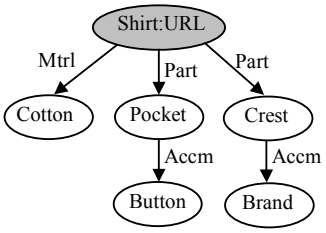

Fig. 3. One candidate resource RDF graph.

The gray nodes represent the respective entries of the two graphs. The arc label "*Mtrl*" is the abbreviation for "Material", while "*Accm*" is for "Accompaniment".

Generally, the query is about a cotton garment with buttoned pouch and brand emblem. To determine whether the candidate resource graph is an appropriate match to it, we need to find out the semantic similarity between them.

3.1 Semantic Similarity

Previous work in (Poole and Campbell 1995) defined three kinds of similarity, i.e. *surface similarity*, *structure similarity* and *thematic similarity*. Surface similarity and structure similarity is the similarity based on the matching of objects and relations respectively, while thematic similarity depends on the presence of particular patterns on objects and relations together. Some related graph matching algorithms, such as Similarity Flooding (Melnik, Garcia-Molina, and Rahm 2002), Cupid (Madhavan, Bernstein, and Rahm 2001) and Anchor-PROMPT (Noy and Musen 2001), also involved, implicitly or explicitly, the idea to unite the linguistic similarity with structural similarity. Hence, we introduce similarity between nodes as well as similarity between arcs and integrate them to construct our similarity between graphs.

Similarity between Nodes. We borrow Sowa's classical definition of *semantic distance* between concept types (Sowa 1984) and simply calculate the similarity between two nodes as 1 minus their distance which cumulates all the distance from each node to their closest common parent. A value depicting the depth information in the ontology is assigned to every single node to help compute the distance.

For example, we want to know the similarity between nodes "*Emblem*" and "*Pocket*". Consulting WordNet, we get the ontology segments as figure 4.

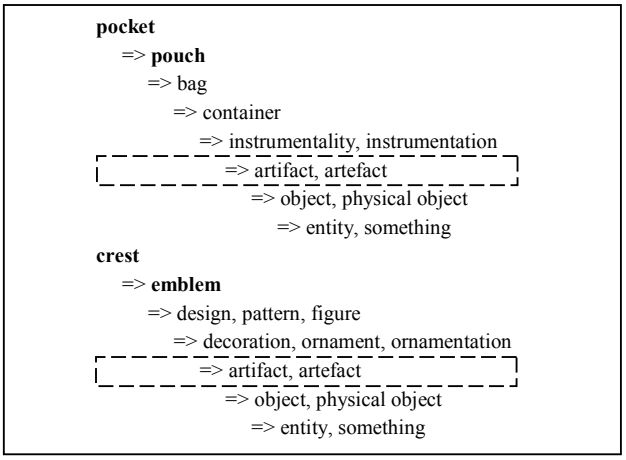

Fig. 4. Ontology segments in WordNet.

"artifact, artefact" is the closest common parent of "emblem" and "pocket". The depth information for "artifact, artefact", "emblem" and "pocket" is respectively 0.0625, 0.0078 and 0.0020. Thus, the distance between "*Emblem*" and "*Pocket*" is (0.0625 - 0.0078) + (0.0625 - 0.0020) = 0.1152. Accordingly, the similarity is 0.8848.

It is important that if the node from query graph subsumes the one from resource graph, the distance between them will be directly set 0. E.g., the distance between "*Pouch*" from query graph and "*Pocket*" from resource graph is 0. This can be explained by the heuristic that a resource category is sure to be considered as an exact match to its superclass category in query.

Similarity between Arcs. We have manually constructed a type hierarchy on arcs according to Sowa's theory about *thematic roles* (Sowa 2000). Therefore, the similarity definition on nodes is transferable to arcs.

In practice, however, we make a simplification to reduce computation complexity. That is, we take similarity as 1 only if the arc from query graph subsumes the one from resource graph; otherwise, the similarity is set 0.

For instance, since none of the arcs "*Mtrl*", "*Part*" and "*Accm*" is subsumed by one another in our relation type hierarchy, the similarity between any two of them is 0.

Similarity between RDF Graphs. To avoid the NP-complete computation of *Maximum Subgraph Matching*, we only compare the nodes and arcs at peer position referring to the entries. Nevertheless, unlike the implementation of OntoSeek, we still try to retain the structure of subgraph. This is carried out through recursive computation in the *similarity formula*:

$$SoG(n_Q, n_R) = w(n_Q, n) \cdot sim_n(n_Q, n_R) + \max_{\substack{for\ all \\ combinations}} \{\sum_j w(n_Q, j) \cdot sim_a(a_Q^j, a_R^j) \cdot SoG(n_Q^{a_Q^j}, n_R^{a_R^j})\}$$

Here, n_Q represents the entry of query graph, while n_R is the entry of resource graph. $SoG(n_Q,n_R)$ is to calculate the similarity between two graphs indicated by their respective entries. $sim_n(n_Q,n_R)$ is the node similarity between n_Q and n_R, whereas $sim_a(a_Q^j, a_R^j)$ is the arc similarity between the jth arc derived from n_Q and its matching arc derived from n_R. $w(n_Q,n)$ and $w(n_Q,j)$ are the weights allocated to the nodes and arcs within current calculation. They are normalized in advance:

$$w(n_Q, n) + \sum_j w(n_Q, j) = 1, \text{ for each subgraph with } n_Q \text{ as its entry}$$

$SoG(n_Q^{a_Q^j}, n_R^{a_R^j})$ represents the recursive calculation of the similarity between the subgraphs with $n_Q^{a_Q^j}$ and $n_R^{a_R^k}$ as their respective entries; $n_Q^{a_Q^j}$ and $n_R^{a_R^k}$ are the nodes that arc a_Q^j and a_R^j point to. $\sum\cdot$ indicates a cumulation operation and **max{·}** selects from all possible combinations the one which contributes the maximum cumulative similarity.

Let us recall the example at the beginning of Section 3. Assume that within every single recursive computation, the entry of graph and its derived arcs share equal weights.

The matching starts with two entries. As "garment" subsumes "shirt" according to WordNet, sim_n(*Garment*, *Shirt*) = 1.

Then we look at the arcs derived from these two entries. Since arc pair with zero similarity will lead to zero product in the second part of the *similarity formula*, we only need to observe arc pairs with non-zero similarity, i.e. identical or subsumptive arc pairs. Easily, we find out that the "*Mtrl*" pair is matching. As the "*Cotton*" nodes are derived from the "*Mtrl*" arcs, we recursively calculate *SoG*(*Cotton*, *Cotton*). Since there are no more arcs derived from the "*Cotton*" nodes, simply we get *SoG*(*Cotton*, *Cotton*) = sim_n(*Cotton*, *Cotton*) = 1.

Go back to the computation of *SoG*(*Garment*, *Shirt*). Now the problem remains how to settle a proper match on the two "*Part*" arcs in each graph. That is to say, we have to make a selection from two possible combinations of derived nodes:

Comb. A: "*Pouch*" – "*Pocket*" and "*Emblem*" – "*Crest*";
Comb. B: "*Pouch*" – "*Crest*" and "*Emblem*" – "*Pocket*".

We need to work out *SoG*(*Pouch*, *Pocket*), *SoG*(*Emblem*, *Crest*), *SoG*(*Pouch*, *Crest*) and *SoG*(*Emblem*, *Pocket*).

Node Similarity	Similarity Value
sim_n(*Pouch*, *Pocket*)	1
sim_n(*Emblem*, *Crest*)	1
sim_n(*Pouch*, *Crest*)	0.7544
sim_n(*Emblem*, *Pocket*)	0.8848
sim_n(*Button*, *Brand*)	0.5049
sim_n(*Brand*, *Button*)	0.5049

Tab. 1. Node similarity values needed in the matching process.

Take *SoG*(*Pouch*, *Pocket*) for example. sim_n(*Pouch*, *Pocket*) = 1. The derived "*Accm*" arcs also matches, so we further investigate *SoG*(*Button*, *Button*). Because no more arcs are derived from node "*Button*", *SoG*(*Button*, *Button*) = sim_n(*Button*, *Button*) = 1. Thus, *SoG*(*Pouch*, *Pocket*) = 0.5sim_n(*Pouch*, *Pocket*) + 0.5sim_a(*Accm*, *Accm*)·*SoG*(*Button*, *Button*) = 0.5*1 + 0.5*1*1 = 1.

Similar to the computation of *SoG*(*Pouch*, *Pocket*), we have:

SoG(*Emblem*, *Crest*) = 1;

SoG(*Pouch*, *Crest*) = 0.6297;
SoG(*Emblem*, *Pocket*) = 0.6949.

Now, we learn that for Comb. A, the second part of the *similarity formula* is 0.25*1*1 + 0.25*1*1 = 0.5; while for Comb. B, the corresponding value is 0.25*1*0.6297 + 0.25*1*0.6949 = 0.3312. Obviously, Comb. A will contribute larger similarity. Thus, by choosing Comb. A, we conclude: $SoG(Garment, Shirt) = 0.25sim_n(Garment, Shirt) + 0.25sim_a(Mtrl, Mtrl) \cdot SoG(Cotton, Cotton)$ + (Comb. A) = 0.25*1 + 0.25*1 + 0.5 = 1. That is to say, the candidate resource graph has a perfect match to the query graph.

Once started, the matching process will not end until all the arcs in query graph have been checked. Arcs in query graph that cannot find its match in resource graph will be regarded as they are mapped to a default relation for we consider it a kind of omission of default values.

3.2 Algorithm Implementation

Given a user query, the following process will be performed to calculate the similarity between each resource RDF graph and the query RDF graph.

```
1  get user query
2  interpret the query to generate query RDF
   graph (with ALPHA)
3  user specifies the entry of graph E; look
   up E in WordNet
4  for (each resource RDF graph indexed by E
   or E's hyponyms in WordNet)
5  { // the beginning of recursive procedure
6    calculate the similarity between query
     and resource entry pair
7    for (each arc derived from the entry of
     query graph paired with each arc derived
     from the entry of resource graph)
8    {
9      calculate the similarity between arc
       pair
10     for non-zero similarity pair, invoke
       the recursive procedure (line #5 to
       #13) with respectively derived node as
       subgraph entries
11     select the best match from all
       possible combinations, and accordingly
       cumulate the similarity between
       entries and the similarity between
       arcs with derived subgraphs
12   }
13 } // the end of recursive procedure
14 rank and orderly return the matching
   results to user
```

Tab. 2. Semantic search process implemented with graph matching algorithm.

Currently a prototype implementing our approach is under development with IBM China Research Lab.

4 Algorithm Evaluation and Discussions

It can be expected that the computation complexity of our algorithm will not reach NP-completeness as the famous problem of *Maximum Subgraph Matching* in this field. Actually, after we introduce the element of "entry of graph" and only compare the nodes and arcs with equal distance to the entry, it is close to a tree-like traversal. Hence without losing generality, suppose that the query graph and the resource graph are both *r*-branch trees of *i* height. To determine all the similarities between subgraph pairs derived from one node pair, there will be at most r^2 times of recursive invocations. Then while making selection from *r*! combinations, we instead employ a kind of Maximum Flow algorithm performed with *r* times invocation of Bellman-Ford algorithm. Bellman-Ford algorithm is r^3 complex, so the cumulative complexity is r^4. Finally, the computation cost between *i* height trees can be analyzed as follows:

Here, *c* is a constant representing the time to calculate the node similarity.

$$\begin{cases} C(i) = c + r^2 C(i-1) + r^4 & i = 1,2,... \\ C(0) = c \end{cases}$$

From the formula group, we may conclude that $C(i)$ is about r^{2i+2}. Generally, when *r* is not too small, n, the number of arcs, will approximate r^i. Therefore, the approximation of $C(i)$ can be converted to n^2r^2. If $r \ll n$, the complexity will be $O(n^2)$. In the worst case, say that the height of the graph equals to 1, i.e. $r = n$, the complexity reaches $O(n^4)$. In a word, our algorithm is confined to polynomial.

The tree-like traversal might be suspected mostly. Here, let us discuss about some graph structures other than tree. *Lattice* structural graphs will be implicitly converted to trees during matching, because the shared nodes will be naturally split along different comparison paths. *Directed cyclic graphs* may lead the split to failure and trap our algorithm into endless loop. Optimistically, graphs of that kind are very rare in our specific domain. Even if existed, they can be detected and handled before the program gets to crash.

5 Related Work

Semantic search (or similarly so-called *content retrieval*) has been raised for years. OntoSeek (Guarino, Masolo, and Vetere 1999), as well as its XML version OSCA (O'Brien 1999), defines the semantic match on isomorphism between query graph and a subgraph of resource graph. In order to avoid NP-completeness of such a computation known as Maximum Subgraph Matching, it instead adopts a classic unification algorithm to fetch corresponding nodes and afterwards check the arc linkage between them. Coincidentally like OntoSeek, SCORE (Aslandogan et al.

1995) also finds out the most similar entities between E-R diagrams before observing the correspondence of involved relationship. We think with that kind of simplification, however, matching on nodes is separate from the organization of subgraph. In contrast, we try to retain subgraph structure in our similarity definition but confine comparison range to mitigate as much the computation cost as possible in matching process.

Graph matching is another independent research topic but highly related to semantic matching. Similarity Flooding (Melnik, Garcia-Molina, and Rahm 2002) relies on the intuition that elements of two distinct models are similar when their adjacent elements are similar. In other words, a part of the similarity of two elements propagates to their respective neighbors. Therefore, they borrow the conception of "flooding" from the field of communication to iteratively compute the similarity until a *fixpoint* appears. Cupid (Madhavan, Bernstein, and Rahm 2001) uses a comprehensive name matching based on synonym tables and other thesauri as well as a new structural matching approach which bases matches on bottom-up traversal and biases matches to schema leaves. Anchor-PROMPT (Noy and Musen 2001) defines the similarity score between two terms in respective ontology as a cumulative score reflecting how often they appear in identical positions along the paths considering all the possible paths between *anchors* (pairs of related terms defined by the user or automatically identified by lexical matching). The algorithm is based on the assumption that developers link the terms in the ontology in a similar manner even if they do not call the terms with the same names.

Though we think our algorithm shares the thought of similarity propagation with Similarity Flooding, shares the idea of bottom-up traversal with Cupid and shares the notion of non-local matching with Anchor-PROMPT, there still exists obvious difference. The difference mainly derives from the distinction in matching objects from models, schemata or ontologies to our instance descriptions. During the design of similarity definition and matching algorithm, they emphasize most on the handling of heterogeneity. However, in our problem domain, heterogeneity is rather rare for both query graph and resource graph are generated according to the same domain ontology. Thanks to this, we are able to concentrate more on the matching of contents in particular patterns.

While we believe that we might contribute some rough ideas to the semantic search problem, we do not claim to have solved it. Testing is necessary to determine the applicability of our algorithm to large-sized graphs and massive resource graph repository. Much experimental and comparative analysis of the algorithm is also needed to demonstrate its strength. There still remain the manipulation problems on matching nested graphs and matching non-isomorphic graphs. We just expect our work to bring more attention and further attempts to the open issue of semantic search.

Reference

Aslandogan, Y. A., Thier, C., Yu, C. T., Liu, C., and Nair, K. R. 1995. Design, Implementation and Evaluation of SCORE (a System for COntent based REtrieval of pictures). In *Proceedings of the Eleventh International Conference on Data Engineering (ICDE)*, 280-287. Taipei, Taiwan.

Berners-Lee, T. 2001. Conceptual Graph and the Semantic Web. *Tim Berners-Lee's Design Issues*. Available at http://www.w3.org/DesignIssues/CG.html

Corby, O., Dieng, R., and Hébert, C. 2000. A Conceptual Graph Model for W3C Resource Description Framework. In *Proceedings of the Eighth International Conference on Conceptual Structures (ICCS'2000)*.

Guarino, N., Masolo, C., and Vetere, G. 1999. OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems* 14(3): 70-80.

Lassila, O., and Swick, R. 1999. Resource Description Framework (RDF) Model and Syntax Specification. *W3C Tech. Reports and Publications*. Available at http://www.w3.org/TR/PR-rdf-syntax

Li, J., Zhang, L., and Yu, Y. 2001. Learning to Generate Semantic Annotation for Domain Specific Sentences. In *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation (K-CAP'2001)*.

Madhavan, J., Bernstein, P. A., and Rahm, E. 2001. Generic Schema Matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB)*. Rome, Italy.

Melnik, S., Garcia-Molina, H., and Rahm, E. 2002. Similarity Flooding: A Versatile Graph Matching Algorithm. In *Proceedings of the 18th International Conference on Data Engineering (ICDE)*. San Jose. CA.

Miller, G. A. 1990. WordNet: An On-line Lexical Database. *The International Journal of Lexicography* 3(4): 235-244.

Noy, N. F., and Musen, M. A. 2001. Using Non-Local Context for Semantic Matching. In *Proceedings of the Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*. Seattle, WA.

O'Brien, P. 1999. OSCA: An Ontology Based Sales Consultant's Assistant. In *Proceedings of the 10th Australasian Conference on Information Systems*.

Poole, J., and Campbell, J. A. 1995. A Novel Algorithm for Matching Conceptual and Related Graphs. In *Proceedings of the Third International Conference on Conceptual Structures (ICCS'95)*, 293-307. Santa Cruz, CA.

Sowa, J. F. 1984. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley.

Sowa, J. F. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, CA: Brooks/Cole.

Zhang, L., and Yu, Y. 2001. Learning to Generate CGs from Domain Specific Sentences. In *Proceedings of the 9th International Conference on Conceptual Structures (ICCS'2001)*.