# Searching the Web: From Keywords to Semantic Queries*

José A. Royo
IEC Department
Univ. of Zaragoza
María de Luna 1
50018 Zaragoza, Spain
joalroyo@unizar.es

Eduardo Mena & Jorge Bernad
IIS Department
Univ. of Zaragoza
María de Luna 1
50018 Zaragoza, Spain
{emena, jbernad}@unizar.es

Arantza Illarramendi
LSI Department
Univ. of the Basque Country
Apdo. 649
20080 San Sebastian, Spain
jipileca@si.ehu.es

## Abstract

*Within the emergent Semantic Web framework, the use of traditional web search engines based on keywords provided by the users is not adequate anymore. Instead, new methods based on the semantics of user keywords must be defined to search in the vast Web space without incurring in an undesirable loss of information.*

*In this paper we propose a system that takes as input a list of plain keywords provided by the user and outputs equivalent semantic queries expressed in a knowledge representation language, that could be used to retrieve relevant data. For the translation task, specialized agents manage a third-party thesaurus and a pool of pre-existing ontologies to obtain the different meanings of the user keywords and discover semantic relationships between them in run-time.*

**Keywords***: Semantic Web, semantics discovering*

## 1 Introduction

One of the most common tasks when someone browses the Web is to search information about some topic. The majority of current Web search engines take as input a list of keywords, which has been proved to be an easy query technique for any kind of users. However, these search engines perform syntactic rather than semantic searches. As current search engines are based on Information Retrieval techniques [4], they obtain links to web pages where those keywords appear. The main problem of their approach is that they do not consider the semantics of the keywords, i.e., what the user actually wants to find.

A few years ago, in the context of Global Information Systems, a different approach was followed [16]. These systems, based on ontologies [12] that describe underlying data repositories, perform a query processing guided by semantics, following a *global-as-view (GAV)* or a *local-as-view (LAV)* approach [8]. However, the strategy followed by these systems is not flexible enough to be used on highly dynamic environments like the Web or its new extension, the Semantic Web.

The *Semantic Web* "*targets to build an extension of the current Web, in which information is given well-defined meaning, better enabling computers and people to work in cooperation*" [5]. Some of the main goals of the Semantic Web are: 1) to allow a semantic query processing, i.e., to retrieve just what the user is looking for, and 2) to automatically discover semantic information among ontologies available in the information system.

In this paper we focus on the previous main goals of the Semantic Web to translate a list of keywords into equivalent semantic queries expressed in a structured language, using a thesaurus and a pool of pre-existing ontologies to obtain the different meanings of such keywords.

Concerning related works we can mention the following. In the Seamless Searching of Numeric and Textual Resources project [14], they use a customized dictionary to disambiguate the concepts used to query the system; however our system uses a general-purpose third-party thesaurus, WordNet, and the context of the user keywords. CUPID [15] and Ontobuilder [11] identify and analize the factors that affect the effectiveness of algorithms for automatic semantic reconciliation, which is a complementary goal to ours: our system matches a set of user keywords with a pool of ontologies, and they match data repositories and ontologies. Finally, GLUE [7] studies the probability of matching two concepts by analyzing the available ontologies, using a relaxation labeling method [19]; however, this approach is not very flexible/adaptable because it analyzes *all* of the ontology concepts, while we use an approach based on a sampling method.

The rest of the paper is as follows. In Section 2 we include an overview of our approach and describe the main

*This work is supported by the CICYT project TIN2004-07999-C02.

agents in the system. Section 3 details how agents can obtain the semantics of a set of user keywords. The matching algorithm and the approach followed to discover semantic relationships and their probability are detailed in Section 4. Section 5 explains the method followed to create semantic queries equivalent to the user keywords, by taking as basis the information obtained in previous steps as basis. Some performance results of the developed prototype are explained in Section 6 and, finally, conclusions and future work appear in Section 7.

## 2 Overview of the System

The main steps followed by the system to translate a set of keywords written by the user into queries expressed in a structured language with a well-defined semantics are the following (see Figure 1):

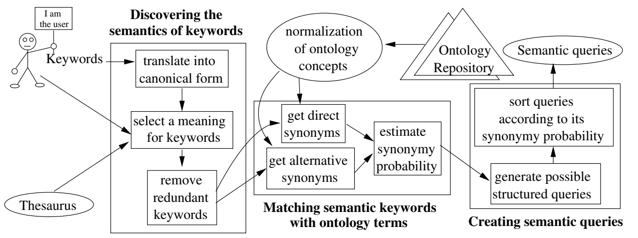

I am the user
Keywords
Discovering the semantics of keywords
translate into canonical form
select a meaning for keywords
remove redundant keywords
Thesaurus
normalization of ontology concepts
Ontology Repository
Semantic queries
get direct synonyms
get alternative synonyms
estimate synonymy probability
Matching semantic keywords with ontology terms
sort queries according to its synonymy probability
generate possible structured queries
Creating semantic queries

**Figure 1. Main steps**

1. *Discovering the semantics of the user keywords*: After performing syntactic normalization, the system obtains the possible meanings of each user keywords using a thesaurus. The output of this task is a list of *semantic keywords*, expressed each keyword as its: 1) description in natural language, 2) synonyms, 3) hyponyms, and 4) hypernyms. After selecting the right meaning of each keyword, redundant keywords are removed.

2. *Matching semantic keywords with ontology terms*: The system estimates the probability of synonymy between a semantic keyword and each term in the ontologies available in the system (such ontologies may have been developed automatically or not). The system samples the hypernyms and hyponyms between the semantic keywords and the ontology terms to increase the performance of the matching algorithm; the size of this sample is estimated using the normal distribution [2].

3. *Creating user queries using a structured language*: The system generates well-defined queries using the ontology terms matched with each semantic keyword. A probability of certainty is defined for each query, based on the probability of the matching relationships used in the previous step. Queries with the highest probability are selected first.

The previous main steps, that will be detailed along the rest of the paper, could be executed in a distributed environment (the user could use a mobile device to query the system, and many computers could store ontologies and/or data repositories). So, the architecture of the system (see Figure 2) has been designed using different specialized agents [18], which manage different kind of knowledge, to minimize and balance the overhead of the different tasks, as described in the following:

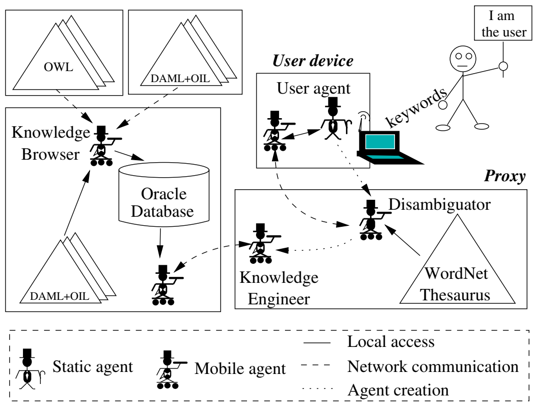


**Figure 2. System architecture**

- *The User agent*: it allows the user to query the system and creates the Disambiguator agent at the fixed network to process such queries. Any further task is performed outside the user device which could have limited capabilities.

- *The Disambiguator mobile agent*: It discovers the semantics of the user keywords; it can interacts with the user (through the User agent), whenever necessary, to help him choose among the different meanings of a keyword. There is one Disambiguator agent for each set of user keywords because the amount of information to analyze is big[1]. It always resides on the proxy computer that provides coverage to the mobile user device, traveling to new proxies as the user moves.

- *The Knowledge Engineer mobile agent*: It is created by the Disambiguator agent in order to discover relationships among the semantic user keywords and terms in ontologies available to the system. It can move to the place where the analyzed ontology pool is stored. It finally returns a list of semantic queries, ordered descendently by synonymy probability, corresponding to the semantics of the user keywords.

- *The Knowledge Browser mobile agent*: it discovers available ontology repositories and creates specialized

[1]The size of the thesaurus used in our prototype is about 40MB.

wrappers [13] to analyze the different languages used to specify such ontologies, like OWL [1]. The analysis of the different ontologies is performed locally if the Knowledge Browser agent can travel where such ontology resides[2]; thus, the analysis can be performed in a distributed and parallel manner (the Knowledge Browser can clone itself). The analyzed ontologies are stored in a database to be shared by all the Knowledge Engineer agents in the system.

## 3 Discovering the Semantics of Keywords

As motivating example, let us suppose that a user wants to buy some publications about thriller films, and therefore he writes the next keywords:

User keywords: *"some publication Book films thriller"*

We detail here the steps given by the Disambiguator agent to discover the meaning of such keywords:

1. *Translating keywords into a canonical representation*: the user keywords can contain noise[3], so the system transforms them into a canonical representation to remove the syntactic heterogeneity that could be introduced by the user. This translation is performed following the next steps: a) *Syntactic normalization*: to rewrite the user keywords in lowercase only; b) *Dehyphenation* [9]: hyphens in labels are removed to improve matching; and c) *Stopterms removal*: this improves the recall and does not adversely affect the precision [10].

   In the example, after this step the user keywords are: *"publication book films thriller"*. The keyword "*some*" is removed as it is a stopterm.

2. *Selecting a meaning for each user keyword*: To obtain the possible meanings of the keywords the system uses WordNet [17] as thesaurus. The system looks up each keyword $k$ in WordNet and stores their definition and semantic properties (synonyms $s_k^{WN}$, hypernyms $H_k^{WN}$ and hyponyms $h_k^{WN}$). As the meaning of user keywords depends *not only on the context but also on the user interpretation*; this task cannot be performed automatically due to the little information available and exploring *all* the possibilities will slow down the process too much. Therefore, in same cases, the system requests the user to select the right meaning from the possible ones found in WordNet, which are ordered by their number of possible meanings, in descending order (the lower the number of meanings, the lower the cost of this task and the fewer the questions to the user).

   In the example, keywords "*publication*", "*book*", "*film*[4]" and "*thriller*" have three, fourteen, seven and one different meanings, respectively, so the user chooses the right one for the first three keywords.

3. *Removing semantically dependent keywords*: At this point the Disambiguator agent knows the wanted meaning of each keyword (they are now *semantic keywords*) so it checks out the semantic relationships existing between them, by consulting the corresponding information in WordNet. This step is very important because the lower the number of keywords, the higher the performance and the better the certainty degree. As result of this step, the user keyword *"publication"* is removed because it subsumes *publication* according to the chosen semantics[5]. .

[2]Mobile agents can travel to computers running an agent context.

[3]Noise is defined as different representations of a word (uppercase and lowercase) and stopterms (common terms such as prepositions and articles) [4].

## 4 Matching Keywords with Ontology Terms

In this section we detail the matching algorithm followed by the Knowledge Engineer agent to create semantic queries according to the chosen semantics of the user keywords. The algorithm matches each semantic keyword with all the terms (classes, properties and instances) in pre-existing ontologies (which can be linked to data repositories).

Our prototype uses the ontology library that is available at the DAML site [6], which contains 282 ontologies about different domains; however, our system works well with any set of ontologies/domains. As these ontologies have been designed by many different people (who used their own semantics), the Knowledge Browser agent must normalize the ontology terms first, by applying the normalization techniques shown in Section 3 and capitalization-based separation techniques [4]. The result is stored in a database available to Knowledge Engineer agents.

### 4.1 General Approach of the Algorithm

The first step of the Knowledge Engineer agent is to discover synonyms of semantic keywords in an ontology pool. For this task it uses a statistical approach to incrementally calculate when a semantic keyword and an ontology term are synonyms, by studying a sample of the hypernym/hyponym relationships of both terms.

Notice that, for example, although a term called *"book"* belongs to an ontology, it could have a different meaning than the user keyword *"book"*. Furthermore, *the formal*

[4]WordNet transforms the keyword "films" into its root "film".

[5]If different meanings are chosen for the keywords, it could happen that $book \not\sqsubseteq publication$.

*descriptions of such terms could be incomparable as they belong to different ontologies/vocabularies*. However, hyponyms and hypernyms of both terms (their *semantic context*) can be compared in order to check their synonymy degree. Thus, a keyword and a term from an ontology always have a certain synonymy probability which depends on the synonymy probability between their hypernyms, and the synonym probability of their hyponyms.

The steps followed by the Knowledge Engineer agent, shown in Figure 3, for each semantic keyword are:

```
 1: for each semantic keyword do
 2:   for each ontology do
 3:     Find direct and candidate synonyms
 4:     for each possible direct and candidate synonym do
 5:       Obtain samples for hyper/hyponyms and possible synonyms
 6:       for each element in samples that is hyponym/hypernym of keyword do
 7:         To apply recursively this algorithm to calculate its synonym degree
 8:       end for
 9:       Calculate synonymy probability of the possible synonym
10:     end for
11:   end for
12: end for
```

**Figure 3. Algorithm to match semantic keywords with ontology terms**

1. *To find the direct and candidate synonyms* (line 3).

   A keyword and an ontology term are *direct synonyms* if such a keyword or any of its synonyms (according to WordNet) has the same writing as that term. This is not enough because they could be homographs. A keyword and an ontology term are *candidate synonyms* if there exists at least one common hypernym or hyponym. So, the system explores the possible synonyms of the keyword with a different writing. In the example, there are seven, three and one synonyms for *"book", "film"* and *"thriller"*, respectively (see Table 1).

2. *To obtain samples* (line 5): If the system would analyze all the hypernymy and hyponymy relationships found then the cost of the algorithm will be NP-hard. Therefore, the system uses a probabilistic approach based on the normal distribution $U(0, 1)$ [2] to decrease the number of comparisons among terms that are performed to estimate the synonymy probability. Therefore, the Knowledge Engineer agent will obtain the sample size, with a certainty degree[6] ($c_d$), for the four sets of terms $H_k^{WN}, h_k^{WN}, H_k^{Ont}$, and $h_k^{Ont}$. The elements of each sample are selected randomly without replacement.

   For the ontologies in the example, the size of the sample (with a 0.95 certainty degree) for the semantic keyword *"book"* is: three hyponyms and one hypernym (from WordNet), and only one hypernym and one hyponym from each related ontology term.

3. *To consider the context of the term* (line 7): the same process is repeated for the hyper/hyponyms of the ontology term that are direct synonyms of hyper/hyponyms of the keyword (according to WordNet).

4. *To estimate the degree of synonymy of the term* (line 9): the system combines the synonymy probability of the hypernyms/hyponyms to calculate the synonymy probability of the term. A certain *threshold* is used to set the minimum synonymy probability to be considered a relevant synonym (in the prototype, threshold=0.950).

   Table 1 shows the synonyms for each semantic keyword in the example and their term type and synonymy probability ($p_s$), when $c_d$ = 0.950.

[6]It usually ranges from 0.900 to 0.975.

| keywd | ontology#term | term type | $p_s$ |
|---|---|---|---|
| book | atlas-publications.daml#Book | class | 0.95320 |
| | docmnt1.0.daml#Book | class | 0.95303 |
| | cs1.0.daml#Book | class | 0.95298 |
| | portal#Book | class | 0.95297 |
| | ka.daml#Book | class | 0.95287 |
| | univ1.0.daml#Book | class | 0.00310 |
| | bibtex.o.daml#book | class | 0.00307 |
| film | CinemaAndMovies.daml#Movie | class | 0.95216 |
| | movienight-ont#Movie | class | 0.95206 |
| | office#Picture | class | 0.00170 |
| thriller | CinemaAndMovies.daml#thriller | instance | 1.00000 |

**Table 1. Synonymy probability**

## 4.2 Optimization of cost: Sampling Terms

The cost of the algorithm shown in Figure 3 is: $O(n, m, d) = K \times (n \times m)^{d+1}$, where $n$ is the number of ontologies, $m$ is the maximum number of possible synonyms of each semantic keyword, and $d$ is the depth of the recursive algorithm. In practice, $d$ can be a small number (see Figure 4.a, in Section 6).

Without samples, the cost of the algorithm would be $O_{\text{NoSamples}}(n, m, r, d) = K \times (n \times m \times r)^{d+1} = O(n, m, d) \times r^{d+1}$, where $r$ is the maximum number of common hypernyms/hyponyms. We cannot assume that $r$ has a low value: in the example, the keyword "*book*" has 81 hyponyms in WordNet. However, the sample for such a keyword has only three terms out of those 81 hyponyms.

Notice that the number of semantic keywords is usually low. Therefore, the cost increases polynomially in the number of semantic keywords and terms of each ontology.

### 4.3 Estimating the Probability of Synonymy

In this subsection we detail how the system calculates the synonymy probability (that we saw in Table 1) for a specific certainty degree[7].

The synonymy probability between a keyword $k$ and a term $t$ of ontology $o$, $p_s(k,t)$, is calculated using the following recursive formula (it considers the depth $d$ of the algorithm):

$$\begin{aligned}
p_s(k,t) &= pD_s(k,t,d) \\
pD_s(k,t,d) &= \begin{cases} \frac{|h_k^{WN} \cap h_t^o| + |H_k^{WN} \cap H_t^o|}{|h_k^{WN}||h_t^o| + |H_k^{WN}||H_t^o|} & \text{if d = 1} \\ c_d \frac{|h_k^{WN} \cap h_t^o| + |H_k^{WN} \cap H_t^o|}{|h_k^{WN}||h_t^o| + |H_k^{WN}||H_t^o|} + \\ + (1 - c_d)\, hH(k,t,d) & \text{otherwise} \end{cases} \\
hH(k,t,d) &= \frac{\sum_{i \in h_t^o \wedge i \equiv j}^{j \in h_k^{WN}} pDs(i,j,d-1) + \sum_{i \in H_t^o \wedge i \equiv j}^{j \in H_k^{WN}} pDs(i,j,d-1)}{|h_k^{WN}||h_t^o| + |H_k^{WN}||H_t^o|}
\end{aligned}$$

where $h_t^o$ is the sampled set of hyponyms of a term $t$ in ontology $o$, $H_t^o$ are its sampled hypernyms, $|A|$ is the cardinal of a set $A$, and $i \equiv j$ means that $i$ and $j$ are synonyms according to WordNet. We denote by $hH(k,t,d)$ the probability of synonymy between the context of the keyword $k$ (its hypernyms and hyponyms) and the context of the ontology term $t$, with depth $d$.

**Theorem 1.** *When the depth increases, the formula to obtain the synonymy probability does not increase, i.e, if* $d_1 \geq d_2$, $pD_s(k,t,d_1) \leq pD_s(k,t,d_2)$.

In other words, or the function finds the synonym probability (with the given certainty degree) or it is not able to find it (such a term could or could not be a good synonym). In any case, *the function never returns high values for "bad" synonyms*. This statement was verified empirically (see Figure 4.a).

## 5 Generating Queries in a Knowledge Representation Language

In this section we detail how the Knowledge Engineer agent generates queries, expressed in a knowledge representation language, that combine the different semantics found in previous steps for the user keywords.

The generation of queries depends on the expressivity of the chosen knowledge representation language, as each synonym will be used to build subexpressions using the kind of constraints (operators) of such a language. In our prototype, we use OWL DL [1] as knowledge representation language because it can be linked to a Description Logics (DL) reasoner [3].
The main steps of this translation process are the following:

[7]In our prototype we make it equal to the certainty degree of the samples, what we call $c_d$.

1. *Generating combinations of synonyms*: For each keyword, the system combines each possible meaning with all the possible meanings of the rest of keywords. Notice that keyword synonyms could have mapped to different types of terms: classes, properties or instances. Thus, the system explores all the combinations to get the semantics of all the keywords, therefore the semantics of the user query will be one of those combinations.

2. *Translating synonyms into class constraints*: a query is a list of class restrictions so, for each given combination of keyword synonyms, and depending on the term type of such synonyms, the system explores the different possibilities where such a term can appear[8]. Each synonym must appear in at least one class constraint. By using a DL reasoner, the Knowledge Engineer agent automatically rejects many inconsistent combinations. For class terms, the system considers all their properties (despite they were not specify by the user as keywords) as they could implicitly link two user keywords

3. *Consistent combination of class constraints*: The class constraints obtained are combined using the possible OWL DL combinations of class expressions. However, by using a DL reasoner, inconsistent combinations are automatically avoided. The remaining class expressions define the class of objects that the user could be interested in, according to the semantics found in the pool of ontologies.

4. *Sorting queries according to their synonymy probability*: By considering the synonym probability of the involved terms, the system obtains a synonym probability ($p_s$) of each query, which depends on the semantics of the DL operators used in such queries to relate terms. For example, $p_s(t_1 \wedge t_2) = p_s(t_1)p_s(t_2)$. Thus, the different queries are sorted by $p_s$ in descending order.

The result of this process, a list of multiontology[9] queries, can be used to access the data repositories underlying such ontologies [16]. Therefore, the semantics of the user query will be the corresponding to one of those queries. The exploitation of the generated queries goes beyond the scope of this paper and it will be subject of future work.

## 6 Performance Analysis of the Prototype

The prototype uses the ontology library on the DAML site [6], that has 282 ontologies, including 67,987 classes

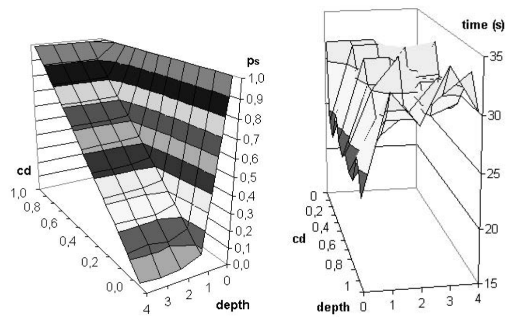
[8]The different DL constraints in OWL DL that can be used to build a query are not included here due to space limitations but can be found in [1]

[9]They could include terms from different ontologies.

and 11,149 properties. The tests were executed on two PCs, wired connected, and the user device was a wi-fi PDA.

The average time of several executions of the prototype shows that the system has a lineal cost in the number of user keywords. From another point of view, the selected user keywords have a minimum impact on the system performance because the hypernyms and hyponyms of the terms are sampled, i.e., the time consumed is the same independently of the user keywords.

In Figure 4.a we show how the algorithm depth and the certainty degree ($c_d$) influence the synonymy probability ($p_s$) obtained between the keyword "book" and a term in an ontology. Notice that the higher confidence in WordNet (high $c_d$), the higher $p_s$; however, we see that the deeper the search, the more confidence in the $p_s$ found, due to the not increasing probability function. Very similar results were obtained for the keyword "film".



(a) (b)

**Figure 4. (a) Synonymy probability ($p_s$) for keyword "book" and (b) time consumed depending on the $c_d$ and depth**

In Figure 4.b we see that, as we sample terms, the increase of depth does not imply an exponential increase in the computing time, and that the computing time is independent of the certainty degree ($c_d$).

## 7 Conclusions and Future Work

In this paper we have presented an approach to translate a semantically ambiguous list of user keywords into unambiguous semantic queries expressed in a language with a well-defined semantics. Our proposal has the following features: 1) The system discovers the possible meaning of user keywords by consulting a thesaurus (WordNet), and helps the user to select the correct meaning; 2) a pool of pre-existing third-party ontologies is used to relate semantic keywords to terms; 3) the system optimizes the number of comparisons among terms by using statistical techniques; and 4) the system generates well-defined queries with the same semantics as the original user keywords.

Our future work is directed to process the obtained queries on a highly dynamic global information system.

## References

[1] G. Antoniou and F. van Harmelen. Web ontology language: OWL. In S. Staab and R. Studer, editors, *Handbook on Ontologies in Information Systems*. Springer-Verlag, 2003.

[2] D. C. M. at al. *Applied Statistics and Probability for Engineers*. Wiley Text Books; 3rd edition, August 2002.

[3] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Pastel-Scheneider. *The Description Logic Handbook. Theory, Implementation and Applications*. Cambridge University Press, 2003.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Adisson-Wesley, ISBN 0-201-39829-X, 1999.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientic American*, May 2001.

[6] DAML. http://www.daml.org, 2000.

[7] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *In The $11^{th}$ International WWW Conference, Hawaii*, 2002.

[8] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the World Wide Web: A survey. In *SIGMOD Record*, 1998.

[9] C. Fox. Lexical analysis and stoplists. In I. W. Frakes and e. R. Baeza-Yates, editors, *Information Retrieval, Data Structures and Algorithms*, pages 102–130. Prentice Hall, Englewood Cliffs, NJ, 1992.

[10] W. Francis and H. Kucera, editors. *Frecuency Analysis of English Usage*. Houghton Mifflin, New York, 1982.

[11] A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *Very Large Databases Journal*, 2005.

[12] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition, An International Journal of Knowledge Acquisition for Knowledge-Based Systems*, 5(2):199–220, June 1993.

[13] J. Hammer, M. Breunig, H. Garcia-Molina, S. Nestorov, V. Vassalos, and R. Yerneni. Template-based wrappers in the TSIMMIS system. In *Proceedings of the Twenty-Sixth SIGMOD International Conference on Management of Data, Tucson, Arizona*, May 1997.

[14] C. Hui-Min. Design and implementation of the agent-based EVMs system. Technical report, Berkley, 2000.

[15] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. *The Very Large Databases Journal*, pages 49–58, 2001.

[16] E. Mena and A. Illarramendi. *Ontology-Based Query Processing for Global Information Systems*. Kluwer Academic Publishers, 2001.

[17] G. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11), nov 1995.

[18] S. Papastavrou, G. Samaras, and E. Pitoura. Mobile agents for WWW distributed database access. *IEEE Transactions on Knowledge and Data Engineering*, 12(5):802–820, 2000.

[19] A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6:420–433, June 1976.