Learning the Semantic Meaning of a Concept from the Web

Yang Yu and Yun Peng

Department of Computer Science and Electrical Engineering,
University of Maryland Baltimore County, Baltimore, MD 21250, USA
{yangyu1,ypeng}@umbc.edu

Abstract. Many researchers have used text classification method in solving the ontology mapping problem. Their mapping results heavily depend on the availability of quality exemplars used as training data. However, manual preparation of exemplars is costly. In this work, we propose to automatically extract text from web pages returned by a search engine. Search queries are formed according to the semantic information given in the ontology. We have implemented a prototype system that automates the entire process (from search query formation to conditional probability calculation) and conducted a series of experiments. We assessed the effectiveness of our approach by comparing the obtained conditional probabilities with human expectations. Our main contribution is that we explored the possibilities of utilizing web information for text classification based ontology mapping and made several valuable discoveries on its usefulness for future research.

Keywords: Semantic web, ontology mapping, text classification, search engine.

1 Introduction

The semantic web is an "extension of the current web" [1], where information is marked up by ontology languages such as OWL and RDF so that it can be understood and processed by programs. However, it is not realistic to assume everyone shares a single ontology. Instead, different organizations may have different ontologies for the same domain, reflecting their designers' own perceptions and conceptualizations of the domain. For example, a course on neural networks may be called "Introduction to Neural Networks" in one university's course ontology but "Introduction to Connectionist Models" in another's. Understanding these two courses actually teaching similar materials will not be a problem to a computer science professor because in the professor's knowledge base, the two course titles have the same or very similar meaning or semantics. However, when programs based on one ontology try to exchange information with programs based on another, problems will happen. This so-called interoperability problem has been known for a long time in software integration, and becomes more acute in the semantic web [2].

One of the approaches to address this interoperability problem is to map concepts defined in one ontology to semantically identical or similar concepts in another. Text classification is a very powerful technique some have suggested for this purpose

Z. Kobti and D. Wu (Eds.): Canadian AI 2007, LNAI 4509, pp. 98–109, 2007.

[8, 9]. However, its success is highly dependent on the availability of text documents that are exemplars of individual concepts in the ontologies. Manually preparing a good number of exemplars for hundreds of concepts is time-consuming and very costly. This greatly reduces the attractiveness of text classification based ontology mapping. To address this difficulty, we propose to automatically retrieve exemplars from the web, the largest information source available. A prototype system has been built based on this idea, which allows us to experiment with different parameters and methods in each step of this approach. A series of experiments have shown encouraging results.

The rest of the paper is organized as follows. Section 2 provides background and motives of this work; Section 3 presents the technical details of this approach and the prototype system; Section 4 describes the experiments and results; Section 5 discusses related works; and Section 6 concludes with suggestions to future research.

2 Background and Motivation

In computer science, an ontology is a set of concepts each of which can have individual members, its own properties, and its relations with other concepts in the set. For example, Fig.1 shows a simple ontology defined in OWL based on [3].

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Class rdf:ID= "CommercialJet"></rdf:Class>
<rdf:Class rdf:ID= "BoeingJet">
  <rdf:subClassOf rdf:resource ="CommercialJet"/></rdf:Class>
<rdf:Class rdf:ID= "AirbusJet">
  <rdf:subClassOf rdf:resource ="CommercialJet"/></rdf:Class>
<rdf:Class rdf:ID= "Boeing-747">
  <rdf:subClassOf rdf:resource ="BoeingJet"/></rdf:Class>
<rdf:Class rdf:ID= "A-380">
  <rdf:subClassOf rdf:resource ="AirbusJet"/></rdf:Class>
</rdf:RDF>
```

Fig. 1. Ontology for CommercialJet in OWL

From this ontology, we know, by the *SubClassOf* property, that Boeing-747 is a kind of Boeing Jet, which itself is a kind of commercial jet. If we define a "made-in" property, we can specify "Boeing-Jets are *made-in* WA". If the ontology also has information that WA is the same as Washington State, and it is *a-part-of* USA, then these data can be easily used by a program to answer questions like "Find all types of commercial jets that are made in the USA".

By defining relations between concepts, we effectively build a web of concepts, potentially a huge integrated database, where information can be shared among applications easily and more complicated reasoning can be supported. The arrival of this semantic web requires ontologies to be developed and shared by many organizations and individuals. However, it is hard to make ontology development a coordinated centralized activity and it is also a fact that people can use different terms for one concept or similar concepts, so different ontologies can be created for the

same domain. For the semantic web to work, it is imperative to relate or map concepts between such ontologies.

Different approaches to ontology mapping have been developed. Manual mappings between large ontologies have been tried in recent years [4, 5]. The mapping is accurate and it can be saved for future use. The problem is that the size of ontologies can be very large and ontologies can keep growing, which requests a huge amount of continuous human efforts in establishing and maintaining the mapping. Consequently more researchers are looking for ways to map ontologies (semi)automatically.

String matching of concept names in two ontologies [6] is an effective alternative. Large amount of information can be processed very quickly and with a high degree of accuracy. For example, "meetingPlace" and "PlaceOfMeeting" can be matched. But matching "Tank" and "Armored Motor Vehicle" would usually involve complicated lexical analysis, and a complete dictionary such as WordNet has to be consulted.

Many researchers choose more powerful methods of machine learning, especially text classification techniques [7, 8, 9]. Usually, text exemplars for each concept or class in a given ontology ($Onto_A$) are manually collected. Then a text classifier is trained using these data. To map a concept C defined in another ontology ($Onto_B$) to some concept in $Onto_A$, exemplars for C need to be collected and classified into the classifier of $Onto_A$. Based on the initial classification results, algorithms such as [7] and [8] can be used to carry out the further steps of ontology mapping. Text classification based ontology mapping is much less time-consuming than manual mapping, and more powerful than string matching, because semantic meanings of apparently different strings can be analyzed by processing information contained in the provided exemplars. Here, the existence of exemplars for each concept and their relevancy to the concept they represent are the key factors to the effectiveness of this approach. However, finding sufficient, high-quality exemplars manually is costly, and is thus the limiting factor of this approach.

The WWW is the richest information resource available anywhere in the world. Collecting text exemplars from the WWW is a promising approach. To assess its effectiveness, we designed a tool to retrieve documents from the web through a search engine and tried a number of different ways to process the documents downloaded before using them for text classification. We tested our tool by actually performing some preliminary ontology mapping experiments with small-scale ontologies.

3 System Design

Here we use $Onto_A$ to refer to the ontology in which we seek a mapping for a foreign concept and use $Onto_B$ to refer to the ontology that provides the foreign concept. The system has the following main components:

1. A *parser* to parse ontology files in OWL format and to form search queries.
2. A *retriever* to drive a web search engine with the queries generated by the parser and to retrieve a specified number of web pages based on the search results.
3. A *processor* to process the raw HTML documents obtained from the retriever to construct text files as exemplars for concepts in the ontologies.
4. A *model builder* to build a probabilistic model for $Onto_A$ from its exemplars using a text classification software. This model becomes $Onto_A$'s classifier.

5. A *calculator* to feed the text exemplar for concepts in $Onto_B$ to $Onto_A$'s classifier, collects classification outputs and calculates conditional probabilities as initial mapping results.

We chose Google as our search engine and Rainbow [10] as our text classification software. The structure of the system is shown in Fig.2.

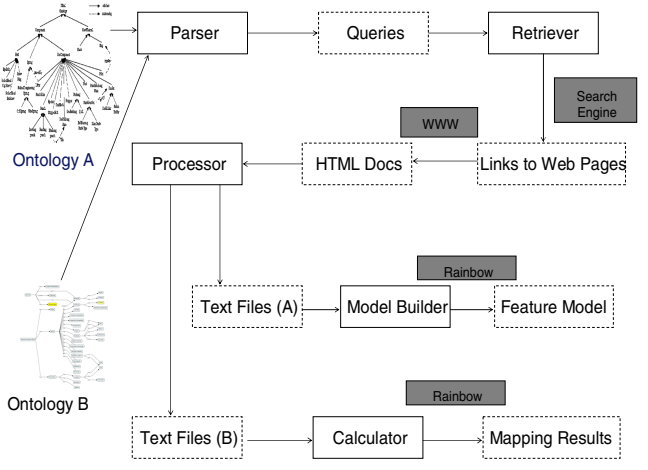

Fig. 2. System components overview

3.1 The Parser

We parse the ontology file to generate search queries for Google. To obtain better results, the query should contain more semantic information than just a class name. Because a word may have multiple senses or meanings, a query consisting of only the words of a concept's name may return web pages based on a more popular meaning of the word, which sometimes is not the particular meaning intended for the concept in the ontology. For example, in an ontology for food with a root class called "food", we may have a concept called apple, which is a subclass of "fruit". If we only use "apple" as the keyword, documents showing how to make an apple pie and documents showing how to use an iPod may both be returned. Apparently, the documents using apple as a computer manufacturing are irrelevant to a subclass of fruit. To avoid this, when forming a query, we use *all the terms on the path from the root class to the class in question* together as a query to send to the search engine. In the apple example, the query would be "food+fruit+apple" instead of "apple" itself alone. By doing so, the number of irrelevant documents returned is greatly reduced. This kind of "word sense disambiguation" by adding additional semantically relevant terms into the search queries can be further extended to include concept's properties. However, it needs to be noted that queries that include too many terms of high specificity (e.g., those in zoology or botany) may lead to very few search results.

3.2 The Retriever

The retriever takes a file containing queries generated by the parser, initiates a connection with a search engine, and sends a query in. It then goes through the search result pages for the query one by one and extracts URLs from each result page. After it collects a pre-specified number of URLs from the search results, it tries to download web pages at these URLs. Currently, only URLs starting with http:// and ending with .html, .htm or / are extracted because other file types, for example, doc or pdf will be difficult for the processor to process. All the HTML files obtained through a query for a particular class are saved in one directory and will be used by the processor to generate exemplars for that class.

For most of the experiments, we retrieved documents using Google as the search engine, because it is the easiest one to be integrated into our system and it is generally considered the best. Although Google provides a programming API to obtain search results, we decided to develop our own retriever program. This gives us the flexibility to experiment with search engines other than Google (for example, Clusty.com) in some of our experiments.

3.3 The Processor

Documents collected by the retriever are HTML files. These raw data have to be processed before being used as exemplars. The processor will remove all HTML tags, image files, script programs, etc. Also removed are hyperlinks, which may contain some useful semantic information, but more often are just links to other irrelevant pages or websites. Since the retriever can easily retrieve a huge amount of relevant documents from the web, we can afford to be more selective in the process.

After the above steps, we have a large number of pure text files for each concept. The processor can perform some optional tasks: keeping only the sentences where a word in the query appears in each text file. Since not every part of a text file is necessarily relevant to the concept in question, this step may help remove irrelevant information and keep only the most closely relevant text. Text files processed with and without this option are both used in our experiments and the results are compared. The processor can also choose to keep paragraphs, rather than sentences in which query words appear.

3.4 The Model Builder

The system takes Naïve Bayes text classification approach to build a probabilistic model for concepts in Onto_A. In a text classification problem, we need to decide among a set of mutually exclusive categories C_1, C_2, C_3 … C_n, to which category a new document d should belong. This can be determined by which category has the greatest posterior probability, given d, i.e., $\max_i(P(C_i \mid d) \mid i = 1, \ldots, n)$, or equivalently, $\max_i(P(d \mid C_i) * P(C_i))$ since only thing that matters here is the ranks among these categories.

Naïve Bayes approximates $P(d \mid C_i)$ as follows. Let d contain m distinct words $d = \{w_1, \ldots, w_m\}$, where w_i is the frequency of the i^{th} word in d. Then assuming that whether a word appears in a category is independent of other words in that category, we have the Naïve Bayes rule

$$P(d \mid C_i) = P(w_1, ..., w_m \mid C_i) = \Pi_{j=1}^{m} P(w_j \mid C_i)$$

Note that the independence assumption does not hold in general. Despite of this, good performance is still achieved. Details of this method can be found in [11].

A Naïve Bayes classifier requires the predefined categories C_1, C_2, C_3 … C_n to be mutually exclusive and exhaustive, so that the probability results can be correct and sum to 1. In our system, classification categories are closely related to ontology concept classes. Our model builder allows one to select concept classes in different ways when forming these classification categories.

For simplicity, this work only considers OWL ontology files that can be viewed as a concept tree based on the subClassOf property. The leaf classes in such a tree are assumed to be mutually exclusive and exhaustive regarding to the root class. By leaf classes, we mean those classes that do not have a subclass. The default behavior of the model builder is to use all the leaf classes in an ontology as the classification categories. Then Rainbow is called to build a probabilistic model for these categories.

Besides the default behavior, the model builder has an option to build a model for each class in Onto_A except the root. Two categories A^+ and A^- are created by the model builder for class A in Onto_A. A^+ is associated with exemplars for that class, and A^- is associated with exemplars for the complement of that class, which are taken from classes that are not A, not A's ancestors nor A's successors in the ontology tree. The model builder then builds a model using the exemplars for the two categories. This option is not applicable to the root class, because the root's complement is empty. For example, the exemplars for "not CAT" in the ontology tree shown in Fig.3 would include those found for all classes except "CAT" and its ancestors, "ANIMAL" and "LIVING_THINGS".

3.5 The Calculator

Rainbow and other naïve Bayes text classifiers tend to produce extreme values (0 or 1) because of the independence assumption. This is certainly good enough if one only wants to get a right classification. However, our purpose is to use the classifier to obtain $P(A \mid B)$ of concept A in Onto_A, given concept B in Onto_B, and hope to use this value as a basis to measure the semantic similarity between A and B. The calculator solves this problem by providing estimates of true conditional probabilities. It works as follows: (1) feeds all exemplars of concept B of Onto_B one by one to Rainbow, which performs classification using the model of Onto_A, (2) keeps records of the classification results for each exemplar, (3) calculate average results grouped by categories in the model as the conditional probability, and (4) write a summary report. It can also perform some additional calculations like estimating conditional probabilities for mapping involving non-leaf classes.

To see how classification results for a concept is averaged, suppose that APC is a class of Onto_B, a weapons ontology. For simplicity, suppose Onto_A, another weapons ontology, has three leaf classes: TANK-VEHICLE, AIR-DEFENSE-GUN, and SAUDI-NAVAL-MISSILE-CRAFT. We build a model using these three classes as classification categories. To calculate the conditional probability given class APC, we classify each exemplar of APC against the model. Suppose we have 200 exemplars of class APC, and the numbers of exemplars giving result of 1 to the three categories are

170, 20, and 10, respectively. Then by taking the average, the conditional probability P(TANK-VEHICLE | APC) = 170 /200= 0.85, P(AIR-DEFENSE-GUN | APC) = 0.1, and P(SAUDI-NAVAL-MISSILE-CRAFT | APC) = 0.05.

4 Experiments and Results

A large number of ontologies of different sizes have been used in many of our experiments. Due to the page limit, we only report some representative experiments, which involved two sets of ontologies. The first involves a small ontology whose structure is shown in Figure 3. We performed text classification between classes within this ontology and also with some foreign classes. The second set, WeaponsA.n3 and WeaponsB.n3, were taken from I^{3}Con2004 [14].

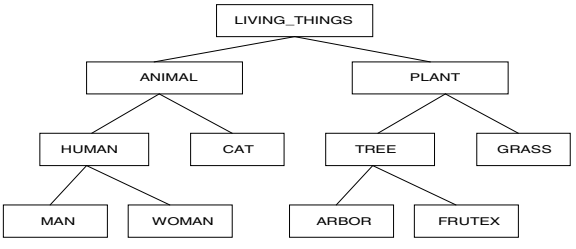

Fig. 3. Structure of LIVING_THING ontology

The system was implemented on Linux and different components developed in perl or Java are glued together by shell scripts. The whole process from parsing, generating queries, to collecting exemplars, building models and calculating results is fully automated.

4.1 Results for Weapons Ontologies

Usually to generate a query for a class, the parser will use all the classes along the path from the root to the class in question. For weapons ontologies, because of their high specificity, to insure that sufficient web pages are returned, we decided to let the parser generate shorter queries, using only the class itself and its parent class.

Onto$_A$, WeaponsA.n3 has more than 60 leaf classes. The model builder ran in default mode, and built a model using these leaf classes as classification categories. The retriever collected 100 exemplars on average for each class. The processor was invoked in two different ways and the results were compared. One is the default mode in which the entire text body of a web page is retained as a pure text exemplar; the other is to only keep sentences containing any of the search keywords as exemplars.

There are 9 classes in WeaponsB.n3 that do not appear in WeaponsA.n3. We try to find a mapping for each of them in WeaponsA.n3. These 9 classes and the manually selected desired mapping leaf classes in WeaponsA.n3 are listed in Table 1.

The conditional probabilities obtained are given in Table 2. For space limitation, here we only list the classes that have the highest probability instead of the complete results over 60 leaf classes for each of the 9 classes. The first column contains classes from WeaponsB.n3. The second and the third columns are the classes in WeaponsA.n3 with the highest conditional probability obtained by using a whole file as an exemplar. The last two columns are results obtained by using only sentences containing keywords as an exemplar.

If we simply judge the mapping accuracy by looking at the class that has the highest conditional probability, it is easy to see that when a whole text document is used as an exemplar, the accuracy is 11% (only LIGHT-AIRCRAFT-CARRIER is correctly mapped). However, the results are improved significantly if we use sentences containing keywords, as exemplars. The accuracy is 56% in this case. There

Table 1. Classes and their desired mappings

Classes from WeaponsB.n3	Desired leaf class mappings
LIGHT-AIRCRAFT-CARRIER	AIRCRAFT-CARRIER
APC	TANK-VEHICLE
SUPER-ETENDARD-FIGHTER	SUPER-ETENDARD
FIGHTER-ATTACK-PLANE	SUPER-ETENDARD
PATROL-WATERCRAFT	PATROL-CRAFT
PATROL-BOAT-RIVER	PATROL-CRAFT
PATROL-BOAT	PATROL-CRAFT
LIGHT-TANK	TANK-VEHICLE
FIGHTER-PLANE	SUPER-ETENDARD

Table 2. Classes with highest conditional probability

New Classes	Whole file	Prob	Keywords Sentences	Prob
LIGHT-AIRCRAFT-CARRIER	AIRCRAFT-CARRIER	0.65	AIRCRAFT-ARRIER	0.57
APC	SILKWORM-MISSILE-MOD	0.46	SELF-PROPELLED-RTILLERY	0.36
SUPER-ETENDARD-FIGHTER	SILKWORM-MISSILE-MOD	0.66	(BALLISTIC-MISSILE) RBM	0.51
FIGHTER-ATTACK-PLANE	SILKWORM-MISSILE-MOD	0.83	(BALLISTIC-MISSILE) RBM	0.38
PATROL-WATERCRAFT	SILKWORM-MISSILE-MOD	0.28	PATROL-CRAFT	0.52
PATROL-BOAT-RIVER	SILKWORM-MISSILE-MOD	0.65	PATROL-CRAFT	0.54
PATROL-BOAT	SILKWORM-MISSILE-MOD	0.51	PATROL-CRAFT	0.66
LIGHT-TANK	SILKWORM-MISSILE-MOD	0.56	TANK-VEHICLE	0.3
FIGHTER-PLANE	AIRCRAFT-CARRIER	0.49	(BALLISTIC-MISSILE) RBM	0.38

are four classes, APC, FIGHTER-PLANE, FIGHTER-ATTACK-PLANE, and SUPER-ETENDARD-FIHTER, whose desired mapping classes do not have the highest conditional probability. We can see that by keeping only sentences containing keywords in an exemplar, noisy information in some web pages can be filtered out, which results in a better classification.

We further looked into those classes that did not get a correct mapping. For class APC, its desired mapping class TANK-VEHICLE has the second highest conditional probability (0.28). We also notice that the one with the highest conditional probability, SELF-PROPELLED-ARTILLERY is also closely related to APC (Armored Personnel Carrier). Text classification method and conditional probability can tell how related two concepts are, but the fact that two concepts are closely related does not mean that they are identical or similar semantically. This case is an example of an interesting problem for future research. For class SUPER-ETENDARD-FIGHTER, its desired mapping class SUPER-ETENDARD also has the second highest conditional probability (0.21). For the other two FIGHTER classes, the results are not good at all. We think one reason is SUPER-ETENDARD is the only leaf node in WeaponsA.n3 that represents a plane (violation of exhaustive assumption for categories). It is possible that it is indeed not a perfect mapping for some plane classes from WeaponsB.n3. To test this, we added a class WARPLANE-OTHER under the class WARPLANE in WeaponsA.n3, containing exemplars retrieved with a search query "WARPLANE+-SUPER+-ETENDARD" and performed the classification process again. Class FIGHTER-PLANE was mapped to its desired WARPLANE-OTHER with the highest conditional probability of 0.41. While class FIGHTER-ATTACK-PLANE still got a wrong mapping. This shows that adding a complement class helps. Moreover, FIGHTER-PLANE is a super class of the other two. The fact that a super class can be correctly mapped will make the mapping of its sub classes easier.

4.2 Results for Living_things Ontology

To obtain further insights of this approach, we conducted the following additional experiments using the "living_things" ontology shown in Fig.3.

1. Calculate P(MAN | HUMAN) and P(WOMAN | HUMAN), both expected to be around 0.5.
2. Given a new, foreign class GIRL, build a model with classes ANIMAL and PLANT as the classification categories and perform classification. If class GIRL is mapped to class ANIMAL, then repeat this process by building a model with Class HUMAN and CAT, and so on.

The system performed these experiments automatically with at most 500 exemplars for each class. Extensive experiments were done with varying parameters. The typical results are reported in Table 3 and 4 (all using sentence-based exemplars).

What is disturbing is that Class CAT has a comparatively high conditional probability given GIRL. This was present during all the experiments we performed for this set of ontology. One reason for this anomaly is that words like human, man,

Table 3. Results of experiment 1

P(MAN | HUMAN)	0.62
P(WOMAN | HUMAN)	0.38

Table 4. Results of Experiment 2

P(ANIMAL | GIRL)	0.76
P(PLANT | GIRL)	0.23
P(HUMAN | GIRL)	0.70
P(CAT | GIRL)	0.30
P(MAN | GIRL)	0
P(WOMAN | GIRL)	1

woman and girl often appear in web pages associated with class CAT because cats have such close relations with human beings (sometimes cat is even used to describe a human). Manual inspection of the exemplars supports this reason.

The "cat" problem shows that even with the best parameters, the exemplars obtained with our system may still be far from perfect. This problem was further confirmed by an additional experiment in which DOG (another domesticated animal) and PYCNOGONID (a kind of sea spider) were added into the ontology as subclasses of ANIMAL. Most of the exemplars of GIRL went to Dog, and none to PYCNOGONID as shown in Table 5.

Table 5. Results with additional classes

P(DOG | GIRL)	0.57
P(CAT | GIRL)	0.03
P(HUMAN | GIRL)	0.40
P(PYCNOGONID | GIRL)	0

We conjecture that, although all exemplars for CAT taken as a whole are closely related to GIRL, it is different at the level of individual exemplars, some are close but others are not. The CAT problem can then be solved if we can separate exemplars that truly reflect the intended semantics of CAT from those that are not. As a first step, we have tried to perform clustering on exemplars of each class in the hope that one of the clusters would contain those truly relevant exemplars. We replaced Google with a clustering search engine Clusty.com that automatically clusters search results based on some text clustering algorithm. Then the largest cluster for each class returned by Clusty.com is used as exemplars. Results are a lot better as shown in Table 6.

We also tried to cluster exemplars obtained by Google search with clustering package in WEKA [15]. Taking the largest cluster does not yield good results this time. These limited experiments indicate that clustering of exemplars is promising in resolving the "CAT" problem, provided we find a way to identify the right clusters.

Table 6. Results by applying clustering on exemplars

P(ANIMAL | GIRL)	0.83
P(PLANT | GIRL)	0.17
P(HUMAN | GIRL)	0.92
P(CAT | GIRL)	0.08
P(WOMAN | GIRL)	0.63
P(MAN | GIRL)	0.37

5 Related Work

Many people have used text classification methods to solve the ontology mapping problem, they include, for example, OntoMapper [9], CAIMEN [7], and GLUE [8]. Although differing in technical details, one thing in common for these methods is that they all require the text exemplars for each concept class be given, and in their experiments, these exemplars are all *manually* collected. To our knowledge, no one has tried to automatically retrieve text exemplars from the web for this purpose.

On the other hand, some researchers in other applications do treat the WWW as a big sampling pool. For example, [16] also uses Google search results to estimate conditional probabilities. For example, P(MAN | HUMAN) would be calculated as the ratio of the number of results for "man" and that for "human+man" (result changes as the number of pages found changes). This method depends on how likely MAN appears on web pages where HUMAN appears; while our method depends on the similarity of pages containing MAN, pages containing HUMAN and pages containing "NOT HUMAN", which brings more semantic information in the contexts and ensures that the probabilities of all the leave classes sum to 1.

6 Conclusions

We proposed to automatically retrieve exemplars from the web for text classification based ontology mapping. We designed and implemented a fully automated system to collect exemplars and calculate conditional probability of two concepts as an initial similarity mapping. The tool can be very useful for ontology mapping tools and frameworks like [7, 8, 9, 12, and 13] and other researches using such a conditional probability [16].

Although our experiment results are mixed, they are in general encouraging and shed lights to the insight of this approach and further work. Two factors probably are most responsible for the less-than-ideal results. The first is the noise in the search results as revealed by the "cat" problem. This may be because that many search results are not really semantically relevant to the keywords, or they are relevant but not semantically close to keywords. The second is that a search is not really a random sampling of the web because all search engines return results according to their own ranking algorithms. How to address these problems and how to best utilize the imperfect exemplars in ontology mapping are the directions for future research.

References

1. Berners-Lee T.: The Semantic Web. Scientific American, 284(5), (2001) 35-35.
2. Wiesman F., Roos N.: Domain Independent Learning of Ontology Mappings. Proc of AAMAS (2004).
3. Ushold M., Menzel C.: Achieving Semantic Interoperability & Integration Using RDF and OWL. http://cmenzel.org/w3c/SemanticInterop.html.
4. Reed S.L., Lenat D.: Mapping Ontologies into Cyc. Proc of AAAI (2002).
5. Niles I., Pease A.: Mapping WordNet to the SUMO Ontology, Proc of the IEEE International Knowledge Engineering conference (2003).
6. Li J.: LOM a lexicon based ontology mapping tool. http://reliant.teknowledge.com/ DAML/I3con.pdf.
7. Lacher M., Groh G.: Facilitating the Exchange of Explicit Knowledge through Ontology Mappings. Proc of the 14th International FLAIRS conference (2001).
8. Doan A., Madhavan J. et al: Learning to Match Ontologies on the Semantic Web. Proc. WWW2002 (2002).
9. Sushama P., Peng Y., Finin T.: A Tool for Mapping between Two Ontologies Using Explicit Information. AAMAS 2002 Workshop on Ontologies and Agent Systems (2002).
10. McCallum A.: Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/~mccallum/bow (1996)
11. Mitchell T.: Machine Learning, McGraw Hill. (1997)
12. Ding Z., Peng Y., Pan R.: BayesOWL: Uncertainty Modeling in Semantic Web Ontologies. Soft Computing in Ontologies and Semantic Web. Springer-Verlag, December (2005)
13. Ding Z., Peng Y., Pan R., Yu Y.: A Bayesian Methodology towards Automatic Ontology Mapping. Proc of AAAI C&O-2005 Workshop. (2005)
14. http://www.atl.lmco.com/projects/ontology/i3con.html.
15. http://www.cs.waikato.ac.nz/~ml/.
16. Perkowitz, M., Philipose, M. et al: Mining models of human activities from the web. In Proceedings of WWW-04. (2004) 573–582.