Finding Appropriate Semantic Web Ontology Terms from Words

Lushan Han[1], Tim Finin[1], Yelena Yesha[1], Robin Salkeld[2], Justin Martineau[1], Li Ding[3], and Anupam Joshi[1]

[1] University of Maryland, Baltimore County, USA
[2] SAP BusinessObjects Academic Research Center, Canada
[3] Rensselaer Polytechnic Institute, USA

Abstract. The Semantic Web was designed to unambiguously define and use ontologies to encode data and knowledge on the Web. Many people find it difficult, however, to write complex RDF statements and queries because doing so requires familiarity with the appropriate ontologies and the terms they define. We describe a system that automatically maps a set of ordinary English words to a set of appropriate ontology terms on the Semantic Web. We use the Swoogle Semantic Web search engine to provide ontology terms and ontology correlation statistics, the WordNet lexical database to resolve synonyms, and a practical three step approach to find the most suitable ontology context as well as appropriate ontology terms.

Key words: Ontology Searching, Data interoperability

1 Introduction

The Semantic Web is realized as a huge graph of data and knowledge. The graph's building blocks consist of literal values, individuals and ontology terms representing classes and properties. Syntactically, an ontology term has two parts: a namespace identifying the ontology defining the term and a local name selecting a term in the ontology. Namespaces resolve ambiguity allowing two terms in different ontologies to share a local name. For example, the class 'Party' can be defined in politics ontology and in a recreational activities ontology. Authoring or querying knowledge on the Semantic Web is difficult because it requires people to select ontologies manually for the concepts they want to use. The term 'Party' is defined in over 400 Semantic Web ontologies known to Swoogle. Moreover, other ontologies define possibly related concepts, with local names like Celebration and Organization. It would be convenient if a user could use natural language words as her vocabulary and a knowledgeable system would suggest appropriate Semantic Web ontology terms.

Given a single English word or short phrase, how do we know which Semantic Web ontology term it should be mapped to? A simple solution is to ask a Semantic Web search engine, like Swoogle [1] or Sindice [2] to return the most highly ranked terms with a corresponding local name. However, people have to

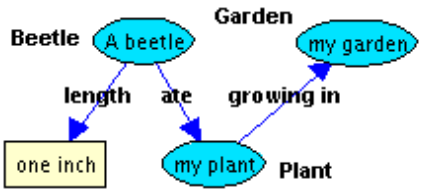

Fig. 1. The semantic graph of a simple sentence "One inch long beetle ate my plant growing in my garden." *Beetle*, *Plant*, and *Garden* are used as classes while *ate*, *growing in* are used as object properties and *length* are used as data property

manually disambiguate the terms and possibly need expand the query with synonyms. Moreover, common use scenarios typically use a set of words rather than a single one. Mapping a word at a time from a set of words while maintaining the consistency of the terms is laborious. One straightforward way to tackle this problem is to find a single Semantic Web ontology such that the whole set of words can all be matched by the terms it defines, where a matching term has a local name which is either the same as the matched word or a synonym of it. The reason it could work is that Semantic Web ontologies are typically small and context-specific. Most define a set of terms for a few or a very limited number of highly related concepts, such as *foaf* for person profile, *geo* for geographical position, and *cc* for license. If a set of words can all be matched by the terms of a context-specific ontology, it is high likely that they have close contexts and consequently the terms should be a good map of the words. For example, consider the short sentence "One inch long beetle ate my plant growing in my garden.", which is represented as a graph in Figure 1. The six words *Beetle*, *Plant*, *Garden*, *ate*, *growing* and *length*, put together, can indicate a specific context. Within this context, the words can be disambiguated. So, if we could find a context-specific Semantic Web ontology that defines the six terms with the corresponding local names, the class 'Beetle' should more likely mean the insect beetle rather than the car beetle and the class 'Plant' should more likely be the organism plant rather than the factory plant.

However, an inherent problem with the previous approach is that the terms may come from multiple ontologies. Instead of a single "Things in Garden" ontology, they may come from three ontologies – 'Beetle', 'Plant' and 'Garden'. Current practice is that people often use terms in different ontologies in authoring an RDF document. Interestingly, this practice bestows upon Semantic Web ontologies a very useful feature – ontologies can be connected by their co-occurrences in existing RDF documents. In this way we can form the ontology co-occurrence network with weighted edges indicating how strongly ontologies are connected to each other. We reduce the weight of very large ontologies such as CYC, WordNet and DBPedia since they define too many concepts. In this network, every ontology has a context which no longer confined to the terms it defines but also include the terms defined in its neighboring ontologies. We

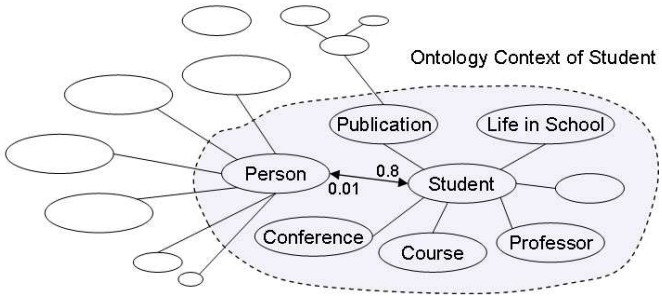

Fig. 2. Student Ontology Context: Ovals represent ontologies. The gray area denotes the ontology context of the student ontology. Some connections in the gray area have been omitted for simplicity. The Person ontology has a conditional probability of 0.8 accompanying the Student ontology while the other way only has a conditional probability of 0.01.

call it ontology context. According to how strongly a neighboring ontology is connected to the central ontology in the perspective of the central ontology, the terms defined in the neighboring ontology are assigned the corresponding weight in the context of the central ontology. The terms in the central ontology have weights of one. There are other ways to assign weights to the terms in an ontology context, but this is simple and, as we will show, effective.

Figure 2 gives an illustration of ontology context. Although shown as undirected, each edge actually consists of two directed edges with different weights. The weight of the directed edge from ontology A to ontology B is the conditional probability of A accompanying B in an RDF document. Even two connected ontologies can have very different ontology contexts. For example, the Student and the Person are connected ontologies. The most weighted terms in the ontology context of Student may come from the Student, Person, Course and other strongly connected ontologies. But all these terms, except for the terms in the Person ontology, should not be the most weighted terms in the ontology context of Person, considering the number of different ontologies that could be connected with the Person ontology.

Finding the most consistent terms for a given set of words is actually the same question as finding the most related ontology context for the words. This can be achieved by finding the ontology context that returns the highest weight sum of its terms that match the input words. For example, suppose we have six words: 'name', 'birthday', 'sex', 'major', 'GPA' and 'take course' where 'name', 'birthday' and 'sex' are defined as terms in the Person ontology and 'major', 'GPA' and 'take course' are defined in the Student ontology. The Student ontology is more related with the six words than the Person ontology because it returns a higher weight sum of the matching terms. The most related ontology context tends to center on the ontology defining the most specific concept

conveyed in the input words. This greatly reduces the ambiguity its words can have.

In finding the most related ontology context, we also need measure how well the terms match the words. Matching a word or short phrase to the local name of an onotlogy term is not trivial. We have to deal with synonyms and morphological variants of English words. Another major issue in realizing this approach is the computation time. For each word in the input set we can generate a term list containing all the semantically related ontology terms. However, the number of all possible term sets composed by picking one term from each different term list could be huge. On the other hand, the number of all possible ontology contexts is equal to the number of ontologies (approximately 25 thousand) discovered by Swoogle. Evaluating each possible term set for each possible ontology context is impractical.

Besides relatedness, popularity is also an important concern in selecting an ontology context. Since anyone can make their own ontology on the Semantic Web, many ontologies overlap in the concepts they define. Choosing the most popular ontology is beneficial because it helps to ensure that the same concept is encoded with the same terms, and also because popularity is the de facto main indicator for measuring trust on the Web.

In this paper, we present a practical approach and a system to find the most suitable ontology context and appropriate ontology terms in the Semantic Web for a set of English words or short phrases.

The remainder of the paper proceeds as follows. Section 2 further introduces our problem by describing the input and output of our system. Section 3 motivates the reader by providing sample use cases for our approach. Section 4 describes relevant related work. Section 5 gives a detailed description of the technical approach, which is followed by the evaluation in Section 6. Finally we conclude the paper with some brief remarks and identify issues for future research.

2 The Input and Output of the system

The motivation of our system is to allow users to use ordinary English words (short phrases) in authoring an RDF graph involving several related concepts. Accordingly, the input is the set of English words in the graph plus specifying each English word is used whether as a class or as a property. Take the graph in Figure 1 as an instance. The input set consists of “Beetle”, “Plant”, “Garden” as classes and “ate”, “growing in”, “length” as propertes. The input doesn’t include the property values, such as “one inch” because class names and property names are identifers for concepts but property values are mostly used to identify instances. Even an instance can be identified we often don’t know which type it plays in the target context since an instance can have many types. By specifying whether an English word works as a class or a property, part of structure information in the graph is included in the input. However, currently we don’t include the structure information about which property word is applied

on which class word. Exploiting this information requires ground knowledge of which property can be applied on which class on the Semantic Web. Having this ground knowledge need a much deeper approach that may require collecting and learning the domain of Semantic Web properties, building global class taxonomy and even ontology mapping whereas our current approach is mainly based on analyzing the co-occurrences of ontologies (concepts) and the terms (cluster of identifiers) defined for each ontology.

Our output is a number (currently twenty) of ranked term sets, where each term set is a possible mapping for the input words. They are ranked by the score given by a goodness function, which considers both consistency and popularity of the results. Within each term set, an input word (short phrase) is matched to a small number (currently three) of candidate Semantic Web ontology terms. They are also ranked, with the leftmost one being the default and have the highest rank score. The reason we supply a few options is that word ambiguity can happen even within a certain ontology context. The ambiguity is caused by the semantic subtlety which is hard to distinguish by only looking at the lexical name of a term. Consequently, we allow people to manually choose an alternative to the default if it is not what they intended and provide term metadata (labels, comments, domain, range, etc) to give more information.

3 Use Case

Our approach can be applied to encode structured and semi-structured data where the input graph with word labels associated with the data is available or can be parsed out. It could support metadata systems like RDFa [3] or Flickr's machine tags [4], which require qualified namespaces whose selection is hard for users who know nothing or little about ontologies.

Here we would like to point out a special usage – translating table data into RDF, which is also our ongoing work. We developed a tool RDF123 [5] which provides users with a graphical interface to create a mapping that captures the entities and relations in a table. This enables easy translation into RDF. With the help of our current system, the labels for entities and relations in the map can be specified in plain english. We can partially automate the process by exploring the table title, the column headings and their associated values in the table. Every column of a table represents a property, whether data property or object property, of a concept in the table. The column headings are accordingly the property names. Although our system doesn't require, as input, the names of involved classes but having them will greatly improve the search accuracy. Class names can be revealed in the table title and the column headings or they can also be infered from values in columns. For example, we may deduce the concept "Capital City" from the column values 'London', 'Beijing', 'Tokyo', and 'Washington D.C.' since "Capital City" is the most specific common type of all these values. Systems like Wikitology [6] and DBpedia [7] can be used to find types for a given individual.

4 Related Work

Our work is related to the problem of word sense disambiguation in computational linguistics [8]. However, we are not working from a single universe of senses derived from a source like WordNet or Roget's Thesaurus, but from a distributed and evolving collection of published ontologies, where many duplicated concepts exist. Moreover, NLP systems mainly focus on word sense disambiguation itself in text while our objective is to encode and retrieve structured data in a way that facilitates data sharing and integration without limiting the expressive power of users.

In Semantic Web area there are two recent works [9, 10] related with translating keyword query to RDF graph query, in which they all face the problem of mapping keywords to semantic entities in their knowledge bases. However, they take two simple ways to deal with the ambiguities. In the SemSearch approach presented by Lei et al. [9], they just find out all the semantic entities matching the keywords. In the work presented by Tran et al. [10], they only consider the first matching semantic entity. Their semantic entities, besides classes and properties, also include individuals and property values. Finding the relations between the matched entities is the highlight of their works. The SemSearch takes a template approach while Tran et al. achieve it by analyzing the subgraphs that cover all the entities in their knowledge base. In this regard our approach is also different from theirs in that the semantic relations in our case are typically given as input by users.

5 Technical Approach

As previously discussed, a brute-force way to find the most suitable ontology context and the terms for a given set of input words is computationally expensive. In this section, we present a practical three-step approach which gives high quality results. According to our definition, any ontology context has a central ontology. Therefore, our first step is to find appropriate central ontologies to use as starting points. Then in the ontology context of each starting point we find the set of most appropriate terms matching the input words. Finally, we rank the term sets acquired from all the starting points using a goodness function considering both consistence and popularity. The top 20 term sets are then presented as output. We start by describing the data structures used and then go through each step in detail.

5.1 Data Structures

Two main data structures in our system are constructed using the December 2008 snapshot of Swoogle's data which has three million RDF documents, about 25K of which are ontologies that define terms. The ontology co-occurrence network is constructed using namespace correlations in all the RDF documents and stored in memory. The other important data structure is the term index, which contains

all *valid* terms discovered by Swoogle. A term is *valid* if and only if it is defined in the ontology identified by the term's namespace. We assume a term to be *valid* if its namespace is not an accessible URI or there does not exist an ontology at the namespace URI. The total number of terms in the term index is about four million. The index is implemented using Lucene [4]. We do a tokenized index on the local name of a term to enable search terms using keywords appearing in the local name. We develop our customized tokenizer that handles the conventions people use to create local names. For an instance, 'PhDStudent' is tokenized as 'Ph D Student'. We also index the type field of a term so that we can retrieve terms according to the given type – class or property. Moreover, we store some selected properties of a term with the index, such as the namespace of the term, the total number of terms being defined in the namespace ontology, the global term rank and several others.

5.2 Step One – Finding Starting Points

Finding ontologies passing the cover rate threshold. A prerequisite of becoming a starting point is that the target ontology must have a cover rate of the input words above a designated threshold. On one hand we hope a good term set could be found in the ontology context of every starting point, which can save the computation time on searching irrelevant ontology contexts. On the other hand we hope the centrals of most suitable ontology contexts should be covered by the starting points. The tradeoff between precision and completeness can be adjusted by the cover rate threshold, which is currently set to 1/5. Before explaining what the cover rate is, we give a few definitions about what is a cover.

Definition 1. *The local name of a term is called semantically-related with an English word as long as one of the word's synonyms, in any inflexed form, exists in the local name.*

Definition 2. *The local name of a term is called semantically-related with a short phrase as long as all the words in the short phrase exist as keywords in the local name without considering their order. In the meantime, we neglect STOP WORDS in short phrase, such as 'a', 'an', 'is', 'be' and etc.*

Definition 3. *An English word (or short phrase) is said to be covered by an ontology if the ontology defines a term whose local name is semantically-related with the English word and whose type (whether class or property) is the same as the given input type of the English word.*

Precisely determining whether two words (short phrases) can have same meaning is very hard primarily due to so many different ways that people can use to express the same meaning [11]. For example, 'take course', 'enrolled in', 'register course', 'has course', and even 'course' can all mean the same thing. So, we coin the *semantically-related* relationship to indicate that the local name of the

[4] http://lucene.apache.org/

term has a potential to share the same meaning with the word (short phrase). The *semantically-related* relationship is different from semantic association. The two words 'horse' and 'rider' may have a significant association but they are not *semantically-related*. The more *semantically-related* relationships are found between a set of input words and a context-specific ontology, the more probable these two have close or much related contexts. The cover rate is defined intuitively as the number of covered words divided by the total number of input words. The reason we make the threshold as 1/5 is based on the observation that the number of ontologies used in an RDF document is typically no more than five and among them one or two ontologies are core ontologies whose terms are more heavily used than others. Accordingly, making the threshold as 1/5 guarantees that at least the core ontologies used in authoring an RDF document or RDF fragment can pass the threshold.

We collect all the ontologies that pass the cover rate threshold in the following way. First, a *semantically-related* term list is generated for each input word or short phrase. Then a hash set is used to contain all the different namespaces appearing in the terms in each *semantically-related* term list respectively. Finally we count the number of the occurrences of a namespace in different hash sets as the number of input words being covered by the namespace ontology.

We use the term index to find the *semantically-related* term list for any given English word or short phrase. First of all we use WordNet [12] to find all the synonyms of a word or short phrase. We define the synonyms of a word as all the words in the synsets of the first two senses of the target word, which are also used as their 1st or 2nd sense. We don't include the words in the synsets of not frequently used senses because they add much more unnecessary ambiguity. Next, for each synonym that is a word we get the stem of it by applying porter stemmer. Then we append wildcard character * to the stem and compose a Lucene query searching on the local names of terms. The query can return us a list of terms which conform to the wildcard pattern. We filter out those terms in the list which don't share the same stem as the synonym has. If the synonym is a short phase, we directly use Lucene AND operator without doing the stemming on each word in the short phrase. Finally we combine all the terms in all the lists populated for each synonym and form the *semantically-related* term list for the input word (short phrase). A term's metadata such as its namespace, its global rank and etc are also read out of the term index and stored with the term in memory. As you can see, the definition of the *semantically-related* relationship is affected by the way that we can efficiently search terms from a word or short phrase.

Ranking ontologies and picking the top 300. We don't use all the ontologies that pass the cover rate threshold as starting points. One reason is that the number of them could be so many that searching on all of them could significantly increase system response time. The other reason is that we try to avoid using big and broad ontolgies (e.g. CYC) as our starting points if sufficiently many other context-specific ontologies qualify. So we rank them and pick the

top 300, which gives real time response in our test computer, as our starting points. The ranking formula is given by:

$$score = \frac{cover\ rate \cdot popularity}{size} \ . \tag{1}$$

where *cover rate* denotes what percent of the input English words (short phrases) is covered by the ontology, *popularity* is the popularity of the ontology (how many times it is used in the existing RDF documents), and *size* is the size of the ontology (the number of terms defined in the ontology). In order to give low priority to broad big ontologies (e.g. CYC) or maliciously made big ontologies, the *size* is set to be inversely proportional to the score.

The starting points, although getting different ranks in this step, will be treated equally in the following steps since we will find appropriate terms in the ontology context of a starting point, no longer limited to the starting point itself.

5.3 Step Two – Finding Appropriate Terms

Finding terms for each input word. we narrow our search to the terms within the ontology context of a starting point. In the previous section we have generated a global *semantically-related* term list for each input word. Here, for each of them we form a local *semantically-related* term list by scanning the global *semantically-related* term list and pick up the terms whose namespaces are of either the starting point ontology or its neighboring ontologies.

Next, what should we require of a term to be an appropriate match of an input word? Above all, the term's local name should be able to share the same meaning with the word. When multiple such terms exist, how strongly the term's ontology is connected with the starting point is one important factor in determining the selection. The other important factor on hand is the global popularity of the term. The global popularity of a term, although not equal to the local popularity, can promote the chance of the term to be selected. More popular terms often means they are more widely accepted. Using popular terms to encode data helps data more visible to others.

However, due to the difficulty of automatically knowing whether the local name of a term can share the same meaning with the input word (short phrase), we have no way to pick up only those terms whose local names are synonyms of the input word (short phrase) from the local *semantically-related* term list and then compare on those terms' other properties to find the most appropriate term. To work around this problem, we develop a similarity measurement for measuring how closely a term's local name is *semantically related* with the input word (short phrase). Then we use a single formula that combines all the three factors to rank the local *semantically-related* term list. The ranking should be able to put the most appropriate terms to the top places although we cannot guarantee the top ones are definitely correct matches because a correct match might not even exist in the ontology context. The ranking formula is given by:

$$match\ score = e^{\alpha \cdot sim + \beta} \cdot corr \cdot term\ rank \cdot g(corr, \frac{1}{pop}) \ . \tag{2}$$

where *sim* is a measurement showing the extend to which the term's local name is *semantically related* with the input word or short phrase, which is mainly based on the lexical similarity calculation, *corr* is the conditional probability that we can see the term's ontology if the starting point ontology is already presented in an RDF document, *term rank* is the global popularity of the term [13] represented as a float number varying in the range $(0, 10^7)$ with a larger number indicating a higher rank, *pop* is the popularity of the starting point ontology, and g is the correlation effect magnification function (explained soon). *sim* is exponential to reflect the dominate role it plays in determining the *match score*. α and β are just two weights.

There are several techniques to measure word similarity, such as Google similarity distance [14]. But they are either based on semantic association or not fast enough to meet our requirement. So we develop a naive method to calculate our own *similarity* , which is a positive value no larger than two to indicate the chance that the local name of the term would share the same meaning with the input word. The method is mainly based on lexical similarity. If the comparison is performed between two words, the similarity score is a baseline score plus the match ratio – the length of the matched text divided by the length of the longer word. The baseline score is for synonym's sake because a synonym may share no lexical similarity with the input word. If the comparison is performed between two phrases, the similarity score is based on the number of matched keywords divided by the total number of keywords in the local name of the term. However, the similarity score between two phrases is properly adjusted so that it falls in the same data range as the similarity score between two words does.

When facing with the terms with the same or close *sim*, only relying on $corr \cdot term\ rank$ to further rank the terms is not enough. A term even with very high rank should be degraded or discarded if its ontology has a trivial correlation with the starting point. It can be a noisy term caused by a misconnected ontology. On the other hand a term even with very low rank can also be desired if its ontology has a very high correlation with the starting point. In other words, the effect of correlation should be magnified when correlation is around 0 or 1. This is done by multipling $corr \cdot term\ rank$ with $g(corr, \frac{1}{pop})$, which is a piecewise function to simulate the curve in Figure 3. The degree of the magnification, i.e. the slopes of the curve around the two ends, is set to be inversely proportional to the popularity of the starting point. This is because the terms defined in a less popular starting point tends to have lower ranks and therefore need more boost to be able to compete with other terms. On the other hand, up-limiting the degree of the magnification helps the terms with very high popularity and considerable correlation have the chance to rank first, such as rdf:type, dc:title, dc:date, geo:lat and cc:License. Sometimes it is hard to make the best choice between a local term with high correlation and a general term with high popularity and considerable correlation but at least we should be able to put all of them to the top places.

The terms with the highest *match scores* are the most appropriate terms that can be matched with the input word (short phrase) within the ontology context of the starting point. We keep a small number (three currently) of the top terms

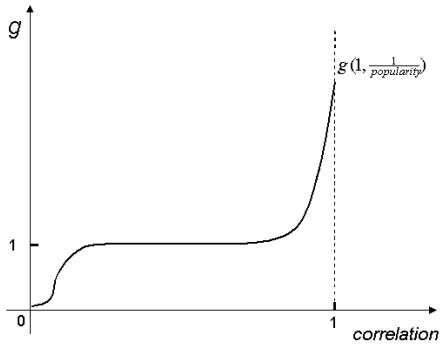

Fig. 3. A curve used to magnify the effect of correlation around 0 and 1

for each input word and put them to a candidate term list respectively. The leftmost one in a candidate term list is default and has the highest rank.

Reducing the number of different namespaces. Can we use the default term in each candidate term list to form the most appropriate term set matching the input words? The answer is not yet. The candidate term list is generated based on the local information of each individual term. However, there is still a constraint that require considering the relationship between terms in the term set. The constraint is that we would like to reduce the number of namespaces used by the terms in the term set. There are two reasons. First, people tend to use vocabulary from as few as possible namespaces to author an RDF document. Secondly, it is beneficial that the most popular terms in different semantics can finally converge on several ontologies but not distributed on many ones in an ontology context. To reduce the total number of namespaces involved, we can try to use the other options in the candidate term list to compose the term set. However, there are more complexities. A term could have too much better a *match score* to be replaced by the other options even it is the only term from its namespace in the term set and its competitor has already many brothers. Thus, the trading of *match score* for the namespace converge must be under a certain condition. We devise a measurement, called converge score, for measuring the converge degree of the namespaces used in a term set, which is given by:

$$converge\ score = \sum_{i=1}^{n} a_i^2 \qquad (\sum_{i=1}^{n} a_i = size)\ . \tag{3}$$

where n is the total number of different namespaces in the term set, a_i is the number of terms using the namespace i in the term set, and $size$ is the number of terms in the term set.

The number of all possible term sets composed by picking one term from each different candidate term list is $size^3$. Therefore finding the optimal term set by evaluating each possible term set is computational expensive. To work

around this problem, we adopt a greedy algorithm – hill climbing. We begin with the set of all default terms, which has the highest *match score* sum, and make the best move every time (a move means a swap of a default term with one of the other two options) until no positive move can be obtained. In order to find the best move, we need evaluate $2 \cdot size$ possible moves every time. The $move\ score_{(i,0)-(i,j)}$ of swapping the term 0 (default term) and term j in the candidate term list i is given by the following formula:

$$\Delta converge\ score\ -\ w \cdot tan(\frac{\pi}{2} \cdot \frac{-\Delta match\ score}{max(match\ score_{(i,0)}, match\ score_{(i,j)})})\ . \quad (4)$$

where $0 \leq i < size$, $1 \leq j < 3$ and w is the weight. The principle of move is that the loss of *match score* can be compensated by the gain of converge score only if the ratio of $match\ score_{(i,j)}$ to $match\ score_{(i,0)}$ is significant. We use tangent function $tan(\frac{\pi}{2}x)$ in the formula because it grows almost linearly when x is under 0.5, grows faster thereafter, and grows abruptly only when x is approaching one. This characteristic can be utilized in constructing the move function that fit with our principle of move.

After the hill climbing procedure terminates, we use the default terms from each candidate term list to construct the default term set which contains the most appropriate terms, within this ontology context, matching the input words.

5.4 Step Three – Ranking the Term Sets

Each starting point can return us a default term set that would be the best mapping in the context of the starting point. In order to find out which term set among them is the most appropriate in the Semantic Web scale for the input set of words, we rank all the default term sets obtained from different contexts according to a goodness function considering both consistence and popularity, which is given by:

$$\frac{pop^{w_1} \cdot tan(cr \cdot (\frac{\pi}{2} - w_2)) \cdot spcr^{w_3} \cdot (\sum_{i=1}^{n} sim_i \cdot corr_i)^{w_4}}{spsize^{w_5}}\ . \quad (5)$$

where sim_i is the measurement showing the extend to which $term_i$'s local name is *semantically related* with the matched input word or short phrase, $corr_i$ is the conditional probability that we can see the $term_i$'s ontology if we already see the starting point ontology in an RDF document, *pop* is the popularity of the starting point ontology, *cr* is the percent of the input words that have matching terms in the ontology context, *spcr* is the percent of the input words that have matching terms belonging to the starting point ontology, *spsize* is the number of terms in the starting point ontology, and w_i are the weights used to adjust the contribution of each factor to the whole formula.

$\sum_{i=1}^{n} sim_i \cdot corr_i$ is the accumulated term weights in the context of the starting point, which also considers how well the term are matched. $(\sum_{i=1}^{n} sim_i \cdot corr_i)^{w_4}/spsize^{w_5}$ alone is a good indicator for measuring the normalized consistency of the target term set. When it is combined with popularity by multiplication, besides using the weights w_1 and w_4 to reach a balance between them

we further introduce $tan(cr \cdot (\frac{\pi}{2} - w_2))$ to promote the term sets that have high cr over those having ordinary cr and introduce $spcr^{w_3}$ to respect the term sets that are more related with their starting points. The big ontologies, although enjoying a high cr and $spcr$, are still inferior to good competitors from small ontologies because of the effect of $spsize$.

Finally we present the top 20 term sets, ordered by their goodness score descendingly, as output to users.

6 Evaluation

Our problem is like an information retrieval task because both aspects find linked data items (schemas or documents) from query words. However, our problem is significantly different from what an IR system does. Ontologies are schemas, while documents are data files encoded in a schema (natural language).

Evaluating Information retrieval systems typically requires ground truth query relevance judgments assessed by humans on document query pairs. Considering the wide variety of queries, this kind of evaluation is very expensive. Instead of using human judgments we use the existing RDF documents known by Swoogle to evaluate our system. We test the system by extracting all the terms (classes and properties) used in an RDF document and use their local names and types as the input words to our system. The original terms in the RDF document compose the target term set. We would like to see how well the top term sets in our suggestions list match the target term set.

We compare our results against a baseline consisting of the globally most popular terms, according to Swoogle's term rank, whose local names and types exactly match the input words. Since the baseline does not need to cope with synonyms, the word ambiguities encountered by the baseline are therefore less than what our system are dealing with. Both our system and the baseline work on the same term index mentioned in Section 5.1.

What we measure is not the correct rate but rather the match rate to the target term set. These two are different because not all RDF documents on the Semantic Web are problem free; terms may be misused, abused, or inferior to other choices. To make the match rate closer to the correct rate, we selected documents that were (1) syntactically well-formed RDF documents, (2) not HTML with embedded RDF, (3) without the string 'test' in their URI, and (4) with local name lengths ranging between zero and 80 characters exclusively. Finally, to prevent test RDF documents from having too many concepts we also required them (5) using no more than 50 different ontology terms.

As of December 2008, Swoogle had more than three million RDF documents. Of these, our selection criteria reduced our test set pool to about 1.4 million entries. We randomly sampled a tenth from this set to use as our final test pool. Since the majority of the dataset documents were foaf and and rss files, the match rates were less convincing since just doing a good job on foaf and rss would yield high accuracy. Consequently, we tested our approach both with and

without the foaf and rss documents to better test the value of our approach. After removing foaf and rss files about 50K entries remained.

Table. 1 shows how well we do against our baseline. We define best@top(n) as the the term set among the top n which has the highest match rate against the target term set. The match rate is simply defined as the number of matching terms divided by the total number of terms in the target term set. There are two ways to decide if a target term is matched: check (i) if the default choice matches and (ii) if any of the three options matches. Consequently, best@top(n) can have two different types. With foaf profiles and rss files, the baseline does a very good job (0.884) because our distribution mirrors actual (popular) usage, which is highly skewed to foaf profiles and rss files. Without foaf and rss, the baseline goes down by 0.084 and the best@top(1) for the default choice also goes down by 0.047. However, a reduction in the match rate does not necessarily mean a reduction of the correct rate. It may reflect an expectation that there are fewer errors or misused terms in foaf and rss documents since they are well-known ontologies and the vast majority of the documents are generated automatically. Figure 4 shows the histograms of the match rate of best@top(1) for the default choice and the baseline to reveal the match rate distributions for the test with foaf profiles and rss documents. Figure 5 shows the same thing but for the test without foaf profiles and rss documents. By comparing the two figures, we can find that best@top(1) has a basically same distribution but the baseline has a much scattered distribution in Figure 5.

Table 1. Average match rate with and without FOAF Profiles and RSS Documents

	With FOAF and RSS		Without FOAF and RSS	
Baseline	0.884		0.800	
best@top(n)	default choice	three options	default choice	three options
1	0.949	0.965	0.902	0.936
2	0.961	0.973	0.929	0.954
3	0.963	0.975	0.935	0.958
4	0.965	0.975	0.938	0.960
5	0.965	0.976	0.939	0.961
6	0.965	0.976	0.940	0.962

7 Conclusion and Future Work

Tens of thousands of Semantic Web ontologies currently exist on the Web and the collection of them determines the vocabulary (terms) that machines can

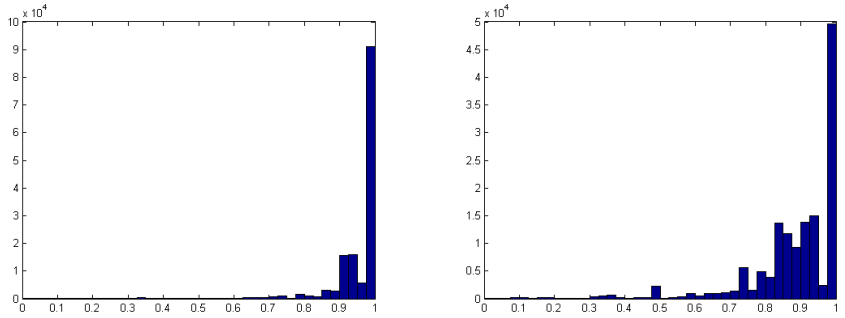

Fig. 4. With foaf and rss: best@top(1) on the left and the baseline on the right

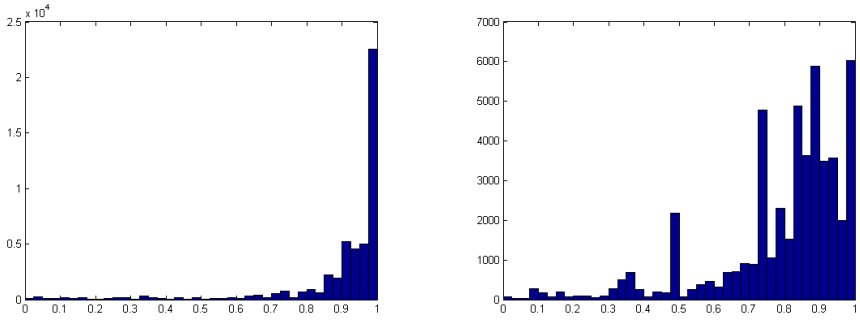

Fig. 5. Without foaf and rss: best@top(1) on the left and the baseline on the right

understand. However, it is very hard or even impossible to let people get familiar with these machine terms. People have their own vocabulary – the words in human languages. Our work is intended to build a bridge between the words and the terms so that people can annotate and query their data in a natural way to them and the corresponding data are stored in RDF data model using the terms. By using the words, people are are provided greater freedom in the variety of data they can annotate. Moreover, through the mapping enabled by the system people can also know what terms are already there and what terms need to be created. This can benefit the development of Semantic Web ontologies and avoid duplicated works. Furthermore, the matching mechanism used in our system determines that the ontologies whose terms are lexicalized in the most common way that people would express English counterpart are more likely to be selected. This can drive ontology development toward meeting people's common conceptualization of the world. One important goal of the Semantic Web is to achieve data integration. We have to first reach consensus at schema level before we can finally integrate instance data. By linking and reusing ontologies, hopefully, there could emerge one or a few dominating ontologies for each unique

concept in the Semantic Web. These dominating ontologies are interlinked with each other to form the standard machine conceptualization of the world. Which ontology will succeed is not determined by any organization but by people and by usage. In future work we plan to apply NLP techniques to better determine if two words (short phrases) can have the same meaning and use ontology information for terms to better understand the semantics of terms.

References

1. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V.C., , Sachs, J.: Swoogle: A search and metadata engine for the semantic web. In: Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management. (2004)
2. Tummarello, G., Delbru, R., Oren, E.: Sindice. com: Weaving the open linked data. In: Proceedings of the Sixth International Semantic Web Conference, Springer (November 2007)
3. Adida, B., Birbeck, M.: RDFa primer 1.0: Embedding rdf in XHTML. W3C working draft 12, World Wide Web Consortium (2007)
4. Schmitz, P.: Inducing ontology from Flickr tags. In: Proceedings of the WWW2006 Collaborative Web Tagging Workshop. (May 2006)
5. Han, L., Finin, T., Parr, C., Sachs, J., Joshi, A.: RDF123: from Spreadsheets to RDF. In: Proceedings of the 7th International Semantic Web Conference (ISWC), Springer (2008) 451–466
6. Syed, Z., Finin, T., Joshi, A.: Wikipedia as an Ontology for Describing Documents. In: Proceedings of the Second International Conference on Weblogs and Social Media, AAAI Press (2008)
7. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: Proceedings of the 6th International Semantic Web Conference (ISWC), Springer (2007) 722–735
8. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: Proceedings of the 33rd annual meeting on Association for Computational Linguistics. (1995) 189–196
9. Lei, Y., Uren, V., Motta, E.: Semsearch: A search engine for the semantic web. In: Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW), Springer (2006) 238–245
10. Tran, T., Cimiano, P., Rudolph, S., Studer, R.: Ontology-based Interpretation of Keywords for Semantic Search. In: Proceedings of the 6th International Semantic Web Conference (ISWC), Springer (2007) 523–536
11. Halevy, A., Norvig, P., Pereira, F.: The unreasonable effectiveness of data. IEEE Intelligent Systems **24**(2) (2009) 8–12
12. Miller, G.A.: WordNet: a lexical database for English. In: Communications of the ACM. (1995) 39–41
13. Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., Kolari, P.: Finding and Ranking Knowledge on the Semantic Web. In: Proceedings of the 4th International Semantic Web Conference (ISWC), Springer (2005) 156–170
14. Cilibrasi, R.L., Vitanyi, P.M.B.: The Google Similarity Distance. In: IEEE Transactions on knowledge and data engineering. (2007) 370–383