Calculating Word Sense Probability Distributions for Semantic Web Applications

Xingjian Zhang and Jeff Heflin
Dept. of Computer Science and Engineering,
Lehigh University
19 Memorial Drive West
Bethlehem, PA 18015, USA
Email: xiz307@lehigh.edu and heflin@cse.lehigh.edu

***Abstract*—Researchers have found that Word Sense Disambiguation (WSD) is useful for tasks such as ontology alignment. Many other Semantic Web applications could also be enhanced with WSD results of Semantic Web documents. A system that can provide reusable intermediate WSD results is desirable. Compared to the top sense or a rank of senses, an output of meaningful scores of each possible sense informs subsequent processes of the certainty in results, and facilitates the application of other knowledge in choosing the correct sense. We propose that probabilistic models, which have proved successful in many other fields, can also be applied to WSD. Based on such observations, we focus on the problem of calculating probability distributions of senses for terms. In this paper we propose our novel WSD approach with our probability model, derive the problem formula into small computable pieces, and propose ways to estimate the values of these pieces.**

***Keywords*-Word Sense Disambiguation; Probabilistic Model; Semantic Web**

I. Introduction

Syntactic matching based on linguistic resources is frequently used in ontology alignment techniques [1]. Magnini et al. [2] have found that understanding linguistic meaning helps find a possible match. Castano et al. [3] rely on linguistic interpretations to find semantic affinity between two concepts. We believe that this work indicates that Word Sense Disambiguation (WSD) plays a very important role in ontology alignment. Besides, it is useful in many other scenarios, such as, but not limited to,

- *Interactive keyword-based ontology search engine*: When a user enters a keyword, the system returns a list of documents that contain the exact keyword, or contain the synonyms or related words of the keyword. The user could then refine the results by specifying which sense the keyword means and the system returns the ones that match only the specified sense.
- *Natural language interface to knowledge bases*: The interface tries to match users' words to the terms in the knowledge base (KB). The input from users and the terms in the KB can be different. The system should understand the meaning of words both from KB and the users, and also check semantic constraints from KB.

In these scenarios, WSD is helpful in two directions. First, it helps find concepts that have different syntactic forms but should be matched. It also avoids aimless expansion of a word to every possible synonym of its multiple senses.

Given so many use cases and applications, we want to develop a reusable component for disambiguating the senses of words in ontologies. WordNet is a common lexicon reference for denoting senses. One common approach is to annotate the target classes by adding mapping axioms to WordNet ontologies [4], which tells what synset in WordNet the concept of each class can be matched to. While the mapping axioms export the senses to Resource Description Framework (RDF) applications, there are two drawbacks. First, there is little work on automatic generation of sense mapping axioms. Second, the class-to-synset granularity makes it sometimes impossible to annotate classes of compound words or a single word in the compound word. In this case, automatic WSD on single words can be useful.

Based on these observations, we address the problem of WSD in Semantic Web documents. In addition to traditional WSD, it involves two new problems: how to decide context by axioms, and how to compute results that are meaningful to subsequent processes. Without a high confidence, a good intermediate result should not be a single top sense. A rank or the scores within the WSD process may be useful, but still be insufficient, because the score and rank do not tell the exact distinctions between candidates. A probability distribution for each ambiguous word provides the most complete information. As probabilistic models have proved successful in many other fields, we propose a novel WSD approach by using a probability model and calculating the distribution as the score results. We will first briefly mention the works that inspire our idea in Section II; then in Section III we introduce how we utilize WordNet; in Section IV we propose our probability model and derive it to computable parts; in Section V we continue with the formula and introduce our approach of estimating the relatedness probability of two senses; in Section VI we present and analyze our preliminary experimental results; and lastly in Section VII we conclude our work and point out future work.

II. Related Work

Banek et al. [5] also stress the importance of WSD in ontology alignment and they recommend WSD as the primary step of ontology integration. They propose their approach of disambiguation on class namesby using the names of the related classes in RDFS axioms as context. However, hey did not consider the names of properties or names of compound words, and only used a limited subset of axioms in the document. They only reported the experimental results on accuracy of top senses.

WSD techniques and many ideas we use in this paper are inspired by many previous traditional WSD works, especially the ones that are knowledge based and exploit information from a given lexicon. One category of these approaches relies on the definition of senses. Lesk [6] first invented the gloss overlap algorithm that calculates the overlap between the definitions of two target words. Banerjee and Pedersen [7] developed the extended gloss overlap method by also considering the glosses of other related senses. Another category of approaches uses semantic similarity measures. For example, Resnik [8] and Jiang and Conrath [9] used the notion of information content from corpus statistical information and calculated the similarity distance between senses. A third category of approaches explores the graph structures and tries to find a lexical chain between target words. Hirst and St-Onge [10] introduced the first computational model of lexical chains and counted the number of times the chain changes direction. A comprehensive review of WSD can be found in [11].

However, most of the previous work define their own scores, based on their own ad hoc heuristics. If we can integrate these heuristics into a more theoretical framework, we may get the combined advantages from different approaches. Also instead of ad hoc scores, probability distributions have clear meanings, and should be easy to reuse. Based on this motivation, we propose our probabilistic approach for WSD in Semantic Web documents.

III. Utilizing WordNet

WordNet is a widely used lexicon for the English language. It groups English words into sets of synonyms called *synsets*, provides short, general definitions, and records the various semantic relations between these synsets. For convenience, we first describe several functions in WordNet.

A *term* is a word or a phrase that can be found in WordNet. A term may have multiple senses, i.e. WordNet provides a function syn which takes a term as input and outputs a set of synsets of this term. Inversely, one could also find the *word forms* of a given synset by the function $wordForm$. A term bound with a synset of it is a *word sense*. The gloss of a synset is the definition of this synset in WordNet. The function $gloss$ takes a synset as input and produces a list of words in the glossary of it as output.

A synset is related to other synsets. WordNet defines a set $\mathcal{E}$ of relation or edge types between synsets. For example, for a noun synset, there are hypernyms (super class), hyponyms (sub class), part holonyms (part of), etc. Note for a given synset the available edges is a subset of $\mathcal{E}$. A function $relEdge$ returns the available edges of a synset, i.e. $relEdge : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{E})$, where $\mathcal{S}$ is the set of all synsets in WordNet, and $\mathcal{P}(\cdot)$ means the power set. Given a synset and an edge type, we could get the synsets related via this type, which is a subset of the set $\mathcal{S}$ of all synsets in WordNet. A function $relSyn$ provides such information, i.e. $relSyn : \mathcal{S} \times \mathcal{E} \rightarrow \mathcal{P}(\mathcal{S})$. Besides the edges defined in WordNet, we add a new type of edge "DONE" which links every synset to *null*. The usage of this edge will be discussed later.

In addition, WordNet provides the statistical information for synsets and terms. A function $tagCount$ tells the frequency of a word sense against a text corpus[1]. From the corpus statistics, we can estimate different types of probabilities. We assume the probability of a given word sense (S, T) is in proportion to the frequency. Thus, the probability that some sense S is the meaning of a given term T is the ratio of the frequency of that word sense against the total frequency of that term, i.e.

$$P(S = s|T = t) = \frac{tagCount(s, t)}{\sum_{s_i \in syn(t)} tagCount(s_i, t)} \quad (1)$$

Similarly, the probability that one tends to use a term T for a given sense S is as follows.

$$P(T = t|S = s) = \frac{tagCount(s, t)}{synTC(s)} \quad (2)$$

where the function $synTC$ of a synset s counts the frequency by summing the tag counts of word senses including all possible variations on the word form of S, i.e.

$$synTC(s) = \sum_{w_i \in wordForm(s)} tagCount(s, w_i) \quad (3)$$

Equations (1) and (2) are probabilities w.r.t. the relations between terms and senses, which reflect our language habits. In some scenarios, we care more about the *concept frequency*, that is, how frequently we meet an instance of a given concept, regardless of how frequently we might use the exact synset of it. For example, the term "Hominidae" is very rare in real world use, but the concept of this term is frequently encountered because it is a generalization of the concept of "human". Thus we define the function cf for counting the concept frequency of a synset S: for nouns it is the size of the set $\mathcal{HS}$ that consists of its direct and indirect hyponyms(subclasses); otherwise it is the same as the $synTC$ of S, i.e.

$$cf(S) = \begin{cases} \sum_{S_s \in \mathcal{HS}} synTC(S_s) & \text{, if } S \text{ is a noun} \\ synTC(S) & \text{, otherwise} \end{cases} \quad (4)$$

Equations (1) and (2) are the *corpus probability formula* we get from statistics against a given corpus. A domain

[1]We set the tag count of a word sense equal to 1 if it is 0 in WordNet.

specific corpus, if available, can provide much better prior knowledge. Equation (1) gives us a prior probability distribution for the meaning of a given term, without considering the context. Our goal is to provide a better estimation of such probabilities with contextual information.

IV. Probability with Context

With contextual information, we can provide better probability distributions for the meaning of terms. In this section, we formally define the problem probabilistically, apply some assumptions, and break down the problem into small computable pieces.

In a Semantic Web document, there are many *URI resources*. A URI resource could be either a class, a property, or an instance that we want to match to something else. Each URI resource has some kind of *associated texts*, such as *rdfs:label*, *rdfs:comment*, or even a parse result of its URI, which are the syntactic information sources. An associated text can be further split into zero or more WordNet *terms*. There might also be words that cannot be found in WordNet, which we ignore here.

To avoid discussion of minutiae, let us simplify the problem by only considering disambiguation within a single ontological document. In real world applications, we might also want to consider a set of ontological documents that contain mappings and other alignment axioms; or even consider the whole KB so that the associated texts of instances are also considered and the RDF triples related to instances are available as clues for disambiguation. Without loss of generality, such a KB, or a set of documents can be viewed as a virtual document.

Now we formally define our problem. Given an ontological document O, and a WordNet term T appearing in the associated text of an RDF resource U,we want to find the probability that this T means the sense S_0, i.e. $P(S_0|O,U,T)$. The condition consists of three parts: the ontology, the URI resource, and the term we want to disambiguate. The discrete random variable S_0 has a domain of $\mathcal{S}$ and stands for the event that T means S_0. All the possible senses are exclusive and exhaustive, thus the sum of all possible senses should be 1, i.e.

$$\sum_{s \in syn(T)} P(S_0 = s|O,U,T) = 1 \tag{5}$$

While T constrains the possibility of values of S_0, O and U are actually the context that have effect on the distribution. Equation (5) defines an ideal probability distribution without information loss, however we have to make some simplifications to estimate it.

Researchers in WSD usually simplify the condition part to some context. The context in theory could be anything, such as very rich structured data. It is basically whatever we want to know from the document in the process of WSD. In traditional WSD, the context is usually defined as a bag of words. We follow this tradition and also define the context in our problem as a bag of terms $W_1, W_2 ... W_n$, i.e.

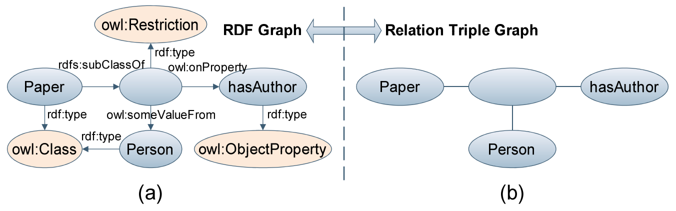

Figure 1. An example of axiom distances. There are three URI resources in this example: Paper, hasAuthor, and Peron. The axiom distance between any two of them is 2.

$$\mathbf{P}(S_0|O,U,T) \approx \mathbf{P}(S_0|T, W_1, W_2, ..., W_n) \tag{6}$$

In traditional WSD, such a bag of words is usually the neighbor words of the target word in the free text document. A window size is set to decide how many words around the target word are included. We shall define the *axiom distance* which is similar to the window size for selecting context.

We first define the set of *relation triples* as all the explicitly stated triples in the document, excluding the ones that use a term from the RDF/RDFS/OWL namespace as its subject or object. Based on these relation triples, we can draw an undirected graph, the *relation triple graph*: each node stands for a unique RDF resource, (which can be a blank node in RDF graphs), and every two nodes, including the properties, that appear together in at least one relation triple are connected with an undirected edge. For any two URI resources, we define the axiom distance as the number of edges in a shortest path connecting them in this relation triple graph [2]. An example is given in Figure 1.

We define the function $context$ as follows. It takes three arguments. The first is the term T we want to disambiguate, the second is the URI resource U of which the label contains T, and the third is an integer that indicates the maximum axiom distance. The output is a bag of words that appears in the labels of URI resources within axiom distance of d of U excluding the ones identical to T. There are two special cases of this function. If we set the axiom distance $d = 0$, it means we only consider the terms that appears in U's label. If $d = \infty$, we consider the labels of every URI resource within the connected RDF graph that contains U. [3]

Banek et al. [5] use a very similar way to find context, but they only consider class names in 4 kinds of axioms, i.e. subclass, superclass, domain and range. Our definition tries to obtain more context words by considering all the axioms and all URI resources with associated texts. It is worth pointing out that our approach of finding context does use

[2]We consider elements within the same parseType Collection to be connected to an anonymous node, and the distance between any two elements in such collections is 2

[3]In practice, there might also be resources that have no axioms in the ontology. The context selection problem in such cases is out of the scope of this paper.

the structure in RDF graphs, but does not use the semantics in it. This makes our approach still a syntactic matching process, and thus is not redundant with subsequent semantic based matching processes.

Once the context is defined, we can further derive the formula in Equation (6) to computable parts as follows.

$$P(S_0|T, W_1, ..., W_n) = \frac{P(W_1, ..., W_n|T, S_0) \cdot P(S_0|T)}{P(W_1, ...W_n|T)}$$
$$= \frac{1}{P(W_1, ...W_n|T)} \cdot P(S_0|T) \cdot \prod P(W_i|S_0, T) \quad (7)$$

We first apply Bayes' rule, then apply the naive Bayes assumption that the occurrence of each W_i in the bag is conditionally independent with others given the disambiguation target word sense, and we have Equation (7). We can interpret this equation as follows. The probability before derivation is the chance that term T has the sense S_0 when a bag of words $W_1, ..., W_n$ co-occur in the context. In the resulting formula, $P(W_1, ...W_n|T)$ is the probability that the bag of words co-occur given that T occurs. Since Equation (5) holds, this part is just a normalization factor for estimating the probability, so we do not need to calculate it. $P(S_0|T)$ is the corpus probability in Equation (1). The product of $P(W_i|S_0, T)$, is the co-occurrence of W_i given the word sense (S_0, T). It can be interpreted as the probability that each term W_i is mentioned when people attempt to define something referred by the target word sense. It tells the relatedness between a term and a word sense. While using WordNet, the condition that a word sense is given is almost the same as the condition that a synset is given, because they provide almost the same information about relations to other synsets or terms except the "antonym" relation. Thus we can use the approximation as follows.

$$P(W_i|S_0, T) = P(W_i|S_0) \quad (8)$$

While diverse approaches of estimating $P(W_i|S_0)$ may be chosen, again we follow the most common one in traditional WSD: the relatedness between synsets. Now we try to transform and relate $P(W_i|S_0)$ to $P(S_y = s|S_0), s \in syn(W_i)$.

The intuition of estimating relatedness between synsets is that more information from WordNet can be utilized if we investigate synsets. $P(W_i|S_0)$ is the probability that W_i is used in the ontology to define S_0. $P(S_y|S_0)$ is the probability that the person thinks of the synset S_y of W_i when attempting to define S_0. We can model the cognitive process with the Bayesian Network reflecting the causal relationships as follows.

$$S_0 \rightarrow S_y \rightarrow W_i$$

The person first has a synset S_0, or say a concept, in mind. This S_0 leads the person to think of another synset S_y, with some probability, in the purpose of defining or explaining this concept in the ontology. At last this S_y is represented with the term W_i by this person. Many synsets can appear given S_0 with some probability, however only the synsets $S_y \in syn(W_i)$ have some probability to cause W_i. Note here S_y is a hidden variable with discrete values. Following the Bayesian Network rules, we have the following equation.

$$P(W_i|S_0) = \frac{\sum_{\forall s} P(S_0)P(S_y = s|S_0)P(W_i|S_y = s)}{P(S_0)}$$
$$= \sum_{s \in syn(W_i)} P(S_y = s|S_0) \cdot P(W_i|S_y = s) \quad (9)$$

$P(W_i|S_y)$ is the corpus probability in Equation (2). The probability $P(S_y|S_0)$ reflects relatedness between synsets.

V. Estimation of Relatedness between Synsets

In order to find the relatedness between two synsets S_0 and S_y, we may need to explore the synset graph in WordNet because S_0 and S_y might not be directly related but are indirectly related via other synsets. Thus, we start the *synset expansion* from the given synset S_0 with the goal of finding chains to S_y. In such expansion, the process that people think of more synsets starting from S_0 is also simulated, thus we propose a model and algorithm that estimates $P(S_y|S_0)$.

A. Synset Expansion Model

We model the expansion as steps of exploration to neighbors in the synset graph from the given synset S_0, with probabilities of deciding which synset to choose at each step. A step of expansion consists of two decisions. First it chooses the WordNet relation type for this step. Some relation types such as "hypernym" have higher probabilities than others, because the connections to other synsets often pass their hypernyms. For example, the synset *cat#n#1* is connected to *paw#n#1* via its hypernym *feline#n#1*. $P(E_1|S_0)$ denotes such probability. In the real world, this reflects the probability that one thinks of a WordNet relation type E_1 when he tries to think about expansion of a synset S_0 in order to define it. The event of deciding a type of WordNet relation edge given the current synset has exclusive and exhaustive values, i.e.

$$\sum_{e \in relEdge(S_0)} P(E_1 = e|S_0) = 1 \quad (10)$$

The second decision of expansion continues with a synset that follows the selected relation edge, and a related synset is selected with some probability, $P(S_1|S_0, E_1)$. This can be viewed as the probability that one thinks of a synset S_1 when he tries to think about a synset related to S_0 with a given type of relation E_1. Similarly, the event of deciding the synset following the relation edge we have chosen also has exclusive and exhaustive values, i.e.

$$\sum_{s \in relSyn(S_0, E_1)} P(S_1 = s|S_0, E_1) = 1 \quad (11)$$

Following Equation (10) and (11), we can derive $P(S_y|S_0)$ at the first step of expansion as follows.

$$\begin{aligned} & P(S_y|S_0) \\ = & \sum_{e \in relEdge(S_0)} P(S_y, E_1 = e|S_0) \quad (12) \\ = & \sum_{e \in relEdge(S_0)} P(S_y|S_0, e) \cdot P(E_1 = e|S_0) \quad (13) \\ = & \sum_{e \in relEdge(S_0)} \sum_{s \in relSyn(S_0, e)} P(S_1 = s, S_y|S_0, e) \cdot P(E_1 = e|S_0) \quad (14) \\ = & \sum_{e \in relEdge(S_0)} \sum_{s \in relSyn(S_0, e)} P(S_y|S_0, E_1 = e, S_1 = s) \cdot P(E_1 = e|S_0) \cdot P(S_1 = s|S_0, e) \\ & \quad (15) \end{aligned}$$

Equation (12) and (14) are derived by marginalization. Equation (13) and (15) can be derived by reforming the conditional probability. Equation (15) is the result of first step expansion. $P(S_y|S_0, E_1, S_1)$ shows that we expand the synset S_0 to its 1st level neighbors, if we can further find the relatedness between S_1 and S_y, we know that S_0 and S_y are somehow indirectly related via S_1. Then we can continue the expansion at the second level between S_1 and S_y.

We define a *chain* after the *l*-th expansion as $C_l = S_0, E_1, S_1, ...E_l, S_l$, $l = 0, 1, ...$. We now show the formula for the (l+1)-th expansion in general case. Note that when $l = 0$, the expansion is the same as above. Deriving the formula is similar to Equation (12)-(15).

$$\begin{aligned} & P(S_y|C_l) \\ = & \sum_{e \in relEdge(S_l)} \sum_{s \in relSyn(S_l, e)} P(S_y|C_{l+1}) \cdot P(E_{l+1} = e|C_l) \cdot P(S_{l+1} = s|C_l, e) \\ & \quad (16) \end{aligned}$$

Equation (16) suggests a recursive algorithm for calculating the relatedness probability for two different synsets. We should also define the[1] exit of recursion, i.e. at some step we should stop expanding the chain C and assign some value to $P(S_y|C)$. We implement it by adding an edge type "DONE" with a small probability at each step. This "DONE" edge expands the last synset S_l of the current chain C_l and links to *null* with probability 1. It indicates we want to force the expansion of this branch to stop and see the direct relatedness between synset S_l and S_y. We further make the assumption that only the last synset in the stopped chain affects the probability.

$$P(S_y|C_l, E_{l+1} = \text{DONE}, S_{l+1} = null) = P(S_y^D|S_l) \quad (17)$$

$P(S_y^D|S_l)$ is the probability that S_y is directly referred given S_l. We use S_y^D to denote the event that S_y is directly related, which also has discrete values. We shall discuss estimation of *direct relatedness* later.

Once we have that defined, we have a finite set of chains $\mathcal{CS}$ to be expanded, then we can rewrite $P(S_y|S_0)$ as follows in a simpler way.

$$P(S_y|S_0) = \sum_{c \in \mathcal{CS}} P(S_y|C = c)P(C = c|S_0) \quad (18)$$

$P(C = c|S_0)$ is the probability of the chain c that starts with S_0. Letting L be the total steps of expansion in c, c_l be the sub chain at the l-th expansion, e_l and s_l be the edge and synset selected at the l-th step, we have

$$P(C = c|S_0) = \prod_{l=0}^{L} P(e_{l+1}|c_l) \cdot P(s_{l+1}|c_l, e_{l+1}) \quad (19)$$

Equation (18) provides another way of understanding the nature of our model. We can also model a Bayesian Network causal graph that leads to Equation (18).

$$S_0 \rightarrow C \rightarrow S_y$$

One intuition of exiting the expansion is that we should stop expansion if the chain is too long. It is unlikely that one synset will remind people of another synset if this is only a distant indirect relation. Mathematically, it means the chain has a very low probability, i.e. $P(C|S_0) < \epsilon$. In this case, we ignore the further expansions that are unlikely to happen, and force the chain to stop expansion by adding a DONE edge with probability 1 after it. i.e.

$$P(S_y|C_l) = P(S_y|C_l, \text{DONE}, null) = P(S_y^D|S_l) \quad (20)$$

Another problem is cyclic chains. Mathematically we have no problem in computation, because cycles make the probability of the chain $P(C|S_0)$ decrease and as the length of the chain approaches infinity, its probability approaches 0. However in reality, we believe people tend to avoid such cyclic thinking in their mind when they try to associate synsets. Thus we remove those expansions that lead to cycles from the possible expansion branches. We shall discuss how we decide possible edges for expansion later in Section V-C.

There are three probabilities we shall estimate: the direct synsets relatedness probability $P(S_y^D|S_l)$, the conditional edge expansion probability $P(E_{l+1}|C_l)$, and the conditional synset expansion probability $P(S_{l+1}|C_l, E_{l+1})$. We first introduce our simple estimation on $P(S_{l+1}|C_l, E_{l+1})$. We assume the probability that C_l is expanded to S_{l+1} in edge E_{l+1} is decided by $cf(S_{l+1})$ in Equation (4), for those synsets that are targets of that edge E_{l+1}. In consistency with Equation (11), we have the normalized estimation.

$$P(S_{l+1}|C_l, E_{l+1}) = \frac{cf(S_{l+1})}{\sum_{s_e \in \mathcal{AS}} cf(s_e)} \quad (21)$$

In the real world, this equation implies that people are more likely to think of synsets that are frequently met. The set $\mathcal{AS}$ is the set of available synsets that are in the set of $relSyn(S_l, E_{l+1})$ but not in the current chain C_l.

B. Estimation of Direct Relatedness between Synsets

$P(S_y^D|S_l)$ tells the direct relatedness between synsets. Here we introduce three different ways to estimate it. A straightforward idea is that we can consider the case that S_y is the same as S_l. Thus we have the first estimation.

$$P_1(S_y^D|S_l) = 1 \text{ iff. } S_y = S_l \text{ , 0 otherwise} \quad (22)$$

However in practice this may not perform well, because it is useful only if we find a chain connecting two synsets S_0 and S_y. Since WordNet does not provide every possible connection between synsets in its synset graph, merely depending on finding explicit chains often fails. For example, there exists no reasonable chain between *person#n#1* and *name#n#1*, which we know are somehow related.

One approach to overcome this problem is using the gloss in WordNet. If some word form T_y of S_y happens to appear in the gloss of S_l, it is evidence that they are related, or mathematically $P(S_y^D|S_l) > 0$. This gives us an approach to estimate $P(S_y^D|S_l)$ with $P_2(S_y^D|S_l)$ based on gloss.

$$P_2(S_y^D|S_l) = \max_{T_y \in wordForm(S_y)} p(T_y, gloss(S_l)) \cdot P(S_y|T_y) \tag{23}$$

There are two parts in the equation. The first part $p(T_y, gloss(S_l))$ is a function that tells the *portion* of T_y in the gloss of S_l, which will be defined soon. The second is the corpus probability in Equation (1), which is the probability that T_y means S_y. We try to match all possible word forms of S_y to the gloss and use the max likelihood. In practice, every word in either the word form or the gloss is stemmed before matching. This allows little variations on the words and even enables the match across different part-of-speech. Thus it increases the chance that matches are found, but may lower the precision.

An easy definition of the function p could be the ratio between the numbers of words, however this makes the estimation of $P_2(S_y^D|S_l)$ biased to common senses, because the senses that have common word forms are much more likely to be matched in the gloss. Thus we use the inverse document frequency(idf) to determine the importance of the words and compute the portion.

$$p(T_y, gloss(S_l)) = \begin{cases} \frac{idfSum(T_y)}{idfSum(gloss(S_l))} & \text{, if } T_y \in gloss(S_l) \\ 0 & \text{, otherwise} \end{cases} \tag{24}$$

We record the document frequency of all the stemmed terms in the gloss of all synsets in WordNet. $idfSum$ is the function that sums the idf of words given a bag of words.

The gloss based estimation helps find the missing relatedness between synsets in WordNet, but still it does not find all. Also we do not want the relatedness between synsets at any time to be 0, a very small probability is better. Note these direct relatedness probabilities are the leaves in every branch of the expansion. If $P(S_y^D|S_l)$ is 0 for every chain, then according to Equation (17) and (18), $P(S_y|S_0) = 0$. If $P(S_y|S_0) = 0$ for every sense S_y of W_i, then according to Equation (9) and (7), $P(W_i|S_0) = 0$, thus $P(S_0|T, W_1, ..., W_n) = 0$. Thus the whole disambiguation is very sensitive to the selection of context, which is not desirable. Thus we define $P_3(S_y^D|S_l)$, the smooth approach

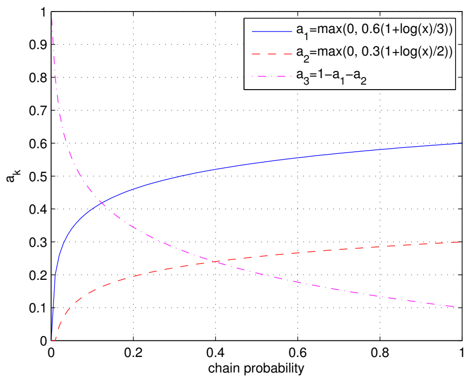

Figure 2. An example group of functions. The x-axis is the chain probability. During expansion, the probability goes down, the values of a_k change from right to left.

of estimating $P(S_y^D|S_l)$.

$$P_3(S_y^D|S_l) = \frac{synTC(S_y)}{\sum_{\forall S} synTC(S)} \tag{25}$$

This simply means the conditional probability is the same as the probability of encountering the synset in the corpus.

With theses three approaches, we have three estimators for $P(S_y^D|S_l)$. We use a simple linear combination method as follows.

$$P(S_y^D|S_l) = \sum_{k=1}^{3} a_k P_k(S_y^D|S_l) \text{ , where} \sum_{k=1}^{3} a_k = 1 \tag{26}$$

There are some heuristics for deciding the weights a_k. The larger a_k is, the better we trust that estimation. Think of one extreme case when $a_3 = 1$. It will always return the same value for the expansion given different S_0, thus it can not disambiguate at all. Making $a_1 > a_2 > a_3$ seems reasonable, because the estimations from 1 to 3 become less reliable. However, as the chain grows longer, the first two approaches gain larger estimation errors, thus become less reliable. So we make these weights the functions of the current chain probability. The initial values of both a_1 and a_2 when the expansion starts decrease to some non-negative values respectively as the chain probability goes down. The functions can be defined very differently. If we assume that the second estimation is more likely to accumulate errors, we can make a_2 decrease with a faster rate as the chain probability goes down. a_3 becomes dominant as both a_1 and a_2 decrease. This can be interpreted as when the chain becomes long enough, the estimation that any synset can be related becomes more accurate. An example group of functions is given in Figure 2. We choose log functions because the chain probability changes approximately exponentially step by step, and we want the change of a_k to be approximately linear w.r.t. the steps.

C. Estimation of Conditional Edge Expansion Probability

$P(E_{l+1}|C_l)$ can be interpreted in the real world as the probability that one thinks of a relation type E_{l+1} given the current chain C_l in mind. We can predefine the weights for different types, that is $weight(E)$ for every relation type E. Then we can estimate $P(E_{l+1}|C_l)$ by normalizing the weights of available edges.

$$P(E_{l+1}|C_l) = \frac{weight(E_{l+1})}{\sum_{E\in\mathcal{ES}} weight(E)} \quad (27)$$

The normalization is required by Equation (10). $\mathcal{ES}$ is the set of available edge types given C_l. It is a subset of edges of the last synset S_l in C_l. To avoid cyclic chains, we prevent edges in $\mathcal{ES}$ from linking S_l to some synset that is already in the chain.

$$\mathcal{ES} = relEdge(S_l) \cap \{E|\exists S \in relSyn(S_l, E) \text{ and } S \notin C_l\} \quad (28)$$

Following Hirst and St-Onge's idea [10] to consider the "number of times the chain changes direction", we can modify Equation (27) to "encourage" the chain to keep its direction, which means the adjacent edge types in the chain are the same. The function aug boosts the weight if the chain keeps the direction.

$$aug(E_l, E_{l+1}) = \alpha > 1 \text{ if } E_l = E_{l+1} \text{ , 1 otherwise} \quad (29)$$

$$P(E_{l+1}|C_l) = \frac{aug(E_l, E_{l+1}) weight(E_{l+1})}{\sum_{E\in\mathcal{ES}} aug(E_l, E_{l+1}) weight(E)} \quad (30)$$

VI. Preliminary Experiments and Discussions

We test our approach on the dblp ontology[4], which is is adapted from the XML schema[5] of the DBLP Computer Science Bibliography[6]. The disambiguation targets are the 39 ambiguous noun terms (i.e. each term that has more than one synset) from *rdf:label* in the ontology. The ground truth is gained by collecting online votes from students who are familiar with the ontology. The base line that our approach is compared to is the corpus probability in Equation (1). We compare two things: (1) the accuracy, i.e. the percentage that the top sense is correct; and (2) the probabilities of the correct sense.

Due to limited space, we cannot demonstrate the results of every possible combination of parameters. Instead, we set the axiom distance level for context $d = 1$, the constant-direction augment factor $\alpha = 1.5$, use a typical stop list for idf and the predefined weights for every type of relation, and use the function group of a_k which combines the direct relatedness estimation as follows.

$$\begin{cases} a_1 &= \max[0, a_{10}(1 - \log_\epsilon p)]; \\ a_2 &= \max[0, a_{20}(1 - \log_{10\epsilon} p)]; \\ a_3 &= 1 - a_1 - a_2 \end{cases} \quad (31)$$

[4]http://swat.cse.lehigh.edu/resources/onto/dblp.owl
[5]http://dblp.uni-trier.de/xml/dblp.dtd
[6]http://www.informatik.uni-trier.de/ ley/db/

Table I
Accuracy Results

(a) $(a_{10}, a_{20}) = (0.5, 0.4)$

$\epsilon =?$	10^{-2}	10^{-3}	10^{-4}	10^{-5}
accuracy	71.8%	76.9%	82.1%	84.6%

(b) $\epsilon = 10^{-4}$

$(a_{10}, a_{20}) =?$	$(0.5, 0.4)$	$(0.9, 0)$	$(0, 0.9)$
accuracy	82.1%	79.5%	79.5%

,where p is the chain probability, a_{10} and a_{20} are the initial values for a_1 and a_2 respectively when $p = 1$.

We compare the accuracy when the chain probability threshold ϵ, and a_{10}, a_{20} change in Table I. In contrast, the accuracy from WordNet top sense is 64.1%. In Table I.(a), we can see the accuracy becomes better when we lower the threshold ϵ, which means we are losing less possible expansion chains. In Table I.(b), we compare the effects by different direct relatedness estimators. When we set a_{10} or a_{20} equal to 0, it means we do not use that estimator of direct relatedness probability ($P_1(S_y^D|S_l)$ and $P_2(S_y^D|S_l)$ respectively). Only relying on one of them can also get good accuracy, but not as good as combining them.

Now we examine the probability distribution of our results. Here we use the setting $\epsilon = 10^{-4}, d = 1, (a_{10}, a_{20}) = (0.5, 0.4)$. We define the *Distribution Candidate Ratio (DCR) Test* as the ratio between the probability of the correct one (from ground truth) and the probability that is the highest *among all other* possible values. This ratio test can be used to evaluate any distribution of discrete value event. If the correct one is not the highest probability in the distribution, this ratio is less than 1 and tells the closeness to candidacy; if the correct one is the highest probability, this ratio is greater than 1 and tells how well the top one is distinguished from the others. In Figure 3 we contrast the ratios of distribution by our approach and WordNet (WN) corpus probability. The terms are sorted by the highest probability output by our approach. From this result we have two findings. First, for most of the cases, our approach has better DCR test result. In comparison to the WN result, our approach either makes the top sense correct, or makes the correct sense more distinguished from the others. Second, we were somewhat surprised that some of the distributions have one dominate probability which is higher than 99%. This could be good for the correct ones showing the confidence of the judgment: the ones with extreme dominant probabilities (also having a very high ratio) are very likely to be correct, because it is usually the sign of finding strong relatedness between the target term and the contexts. For example, term #39 "publication" has the highest dominant probability, because most of the context around it are all related to the correct sense (since the ontology is about publications).

However, there are also exceptions that our result is worse than WN in the ratio test. Term #34 is "master". The correct sense is *master#n#8* which means "someone who holds

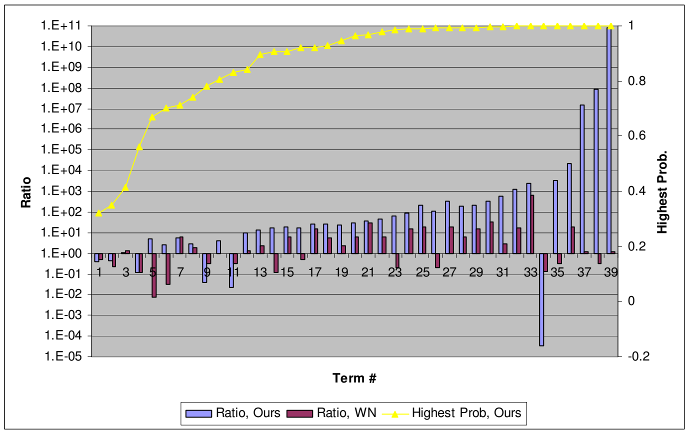

Figure 3. An example ratio of distribution between the correct one and the other top one. Terms are sorted by the highest probability output by our approach.

a master's degree from academic institution", and the top one in our result is *master#n#5* which means "an original creation (i.e., an audio recording) from which copies can be made". The context words, such as "publication", are found related to *master#n#5*, because they are hyponyms (sub classes) of *creation#n#1*. On the other hand, there is little relatedness between context and *master#n#8* found by our approach. However, by humans, it is clear that the word "academic" in the gloss shows some relatedness to context words. Term #9 and #11 are both "title" appearing at different places in the ontology. A similar problem occurs with them. Our approach cannot find the relatedness between the correct sense *title#n#2* and context. However in its gloss, the phrase "literary composition" can be easily related to context words such as "publication" by humans. So all these mistakes our approach makes shows that our challenge is to find efficient ways of estimating the relatedness between synsets, as in Section V-B, and also to minimize the impact of errors.

Currently we have not tried to optimize execution time. The whole process can be very time-consuming when ϵ is small and d is large. The average time for each pair of synsets in $P(S_y|S_0)$ is 278 ms for $\epsilon = 10^{-3}, d = 1$.

VII. Conclusion

Based on the observation that WSD is a common and important task in the Semantic Web, we examined the problem on making the WSD results meaningful and reusable. In this paper, we propose a novel approach for WSD with our probability model. We first construct our probability model of the WSD task, and derive a formula for calculating the sense distribution, and then propose approaches of estimating each term in this formula. Our preliminary experiments show our approach can achieve a 84% accuracy and also make the correct senses distinguished from other senses in the result distribution. As for future work, we think our framework is open to many potential improvements. For example, currently we predefine the weights of edges manually, but heuristic algorithms or machine learning approaches could help generate better weights. In addition, as in our analysis on the experiments, we will try to find other estimators for the direct relatedness probability between synsets. We will also test our approach on more data sets, and research for the evaluation methods for probability distributions.

References

[1] P. Shvaiko and P. Shvaiko, "A survey of schema-based matching approaches," *Journal on Data Semantics*, vol. 4, pp. 146–171, 2005.

[2] B. Magnini, L. Serafini, and M. Speranza, "Linguistic based matching of local ontologies," in *Working Notes of the AAAI-02 workshop on Meaning Negotiation. Edmonton.* AAAI, AAAI Press, 2002.

[3] S. Castano, A. Ferrara, and S. Montanelli, "H-match: an algorithm for dynamically matching ontologies in peer-based systems," in *SWDB*, 2003, pp. 231–250.

[4] M. V. Assem, A. Gangemi, and G. Schreiber, "Conversion of wordnet to a standard rdf/owl representation," in *Proceedings of LREC 2006*, 2006.

[5] M. Banek, B. Vrdoljak, and A. M. Tjoa, "Word sense disambiguation as the primary step of ontology integration," in *Proceedings of the 19th international conference on Database and Expert Systems Applications.* Berlin, Heidelberg: Springer-Verlag, 2008, pp. 65–72.

[6] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone," in *SIGDOC '86: Proceedings of the 5th annual international conference on Systems documentation.* New York, NY, USA: ACM, 1986, pp. 24–26.

[7] S. Banerjee and T. Pedersen, "Extended gloss overlaps as a measure of semantic relatedness," in *International Joint Conference on Artificial Intelligence*, 2003, pp. 805–810.

[8] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *International Joint Conference on Artificial Intelligence*, 1995, pp. 448–453.

[9] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," *Proceedings of the International Conference on Research in Computational Linguistics*, 1997.

[10] G. Hirst and D. St-Onge, "Lexical chains as representations of context for the detection and correction of malapropisms," in *WordNet: An electronic lexical database*, C. Fellbaum, Ed. Cambridge, MA: The MIT Press, 1998, pp. 305–332.

[11] R. Navigli, "Word sense disambiguation: A survey," *ACM Comput. Surv.*, vol. 41, no. 2, 2009.