WebSim: A Pathway to Unveiling Term Relationships using a Web Search Technology

Seokkyung Chung[1⋆], Jongeun Jun[2⋆⋆], and Dennis McLeod[2]

[1] Yahoo! Inc., 2821 Mission College Blvd, Santa Clara, CA 95054, USA
[2] Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA
schung@yahoo-inc.com, [*jongeunj*, *mcleod*]*@usc.edu*

Abstract. We present WebSim (Web-based Similarity metric), whose feature extraction and similarity model is based on a conventional Web search engine. By utilizing the search engine, we can obtain the freshest content for each term that represents the up-to-date knowledge on the term. In comparison with previous text mining approaches that use the certain amount of crawled Web documents as corpus, our method is less sensitive to the problem of data sparseness since we access as much content as possible using the search engine. We also present a method on how to deal with ambiguous terms for the similarity computation. Moreover, we show how existing ontologies can be modified, and demonstrate the characteristics of WebSim by coupling with WordNet. Experimental results show that WebSim can uncover topical relations between terms that are not shown in conventional concept-based ontologies.

1 Introduction

One of the critical issues in intelligent information management is how to represent and extract semantic meanings from information contents. This issue has been explored in diverse research areas including artificial intelligence, information retrieval, natural language processing, etc. The widely used approach to addressing this problem is to utilize ontologies to express semantic knowledge (collections of key concepts and terms along with their inter-relationships) to provide an intelligent information map. For example, query expansion based on ontologies can improve retrieval accuracy when users provide irrelevant keywords (due to their broad and vague information needs or unfamiliarity with the domain of interests).

One of the main bottlenecks in ontology-based approaches is how to build them. Although there exist hand-crafted ontologies such as WordNet [13] or CYC [8], these general-purpose ontologies are less useful for many specific domains (e.g., scientific or engineering applications) due to the lack of capability in expressing the domain of interests. Thus, it is essential to build ontologies

⋆ This research was conducted when the author was at University of Southern California.

⋆⋆ To whom correspondence should be addressed.

that can characterize given applications. Domain specific ontologies are useful for diverse areas. For example, in e-commerce, ontologies can be utilized to customizing a system to a user's preferences. With the popularity of sellers' product recommendation to their customers based on purchasing patterns, ontologies can be effectively used for modelling customers' behavior and users' profiles.

However, although ontology-authoring tools have been developed in the past decades [16], manually constructing ontologies whenever new domains are encountered is an error-prone and time-consuming process. In addition, since ontologies must evolve with time as new concepts or terms appear, it is essential to maintain existing ontologies up-to-date. Therefore, integration of knowledge acquisition with data mining, which is referred to as ontology learning, is necessary. In particular, computation of the similarity between terms is at the core of ontology learning problems. Thus, we focus our attentions on computing similarity between terms.

As the Web continues to grow as a vehicle for the distribution of information, the vast amounts of useful information can be found on the Web. Given this wide availability of knowledge on the Web, we present WebSim (Web-based Similarity metric), whose feature extraction and similarity model is based on a conventional Web search engine.

Figure 1 illustrates the main parts of the proposed framework. In the data gathering component, a term of interest is issued as a query to a Web search engine, which returns a set of the top most relevant Web pages with respect to the term. In order to extract relevant features for the term, the retrieved Web pages are preprocessed by standard IR tools. Based on the features, a vector-space based similarity model can be built and utilized to maintain ontologies up-to-date. An information delivery component can present an answer (e.g., in terms of topic detection and tracking or keyword-based retrieval coupling with ontologies) in response to a user request.

As many thousands of Web pages are published daily on the Web, the Web reflects and characterizes current trend of knowledge. The proposed approach takes advantage of state of the art in Web search engines. For example, for each term, we can always obtain the freshest content, which represents the up-to-date knowledge on this term. Although previous text mining crawls large amount of Web pages for feature extraction, since the crawled contents are just snapshot of the entire Web, it still suffers from a data sparseness problem.

Moreover, we present how to deal with ambiguous terms for the similarity computation, which is one of the difficult problems in a Web search. We also show how ontologies can be restructured or enriched, and demonstrate the characteristics of WebSim by coupling with WordNet. One of the main problems with concept-based ontologies is that topically related concepts and terms are not explicitly linked[3]. That is, there is no relation between *Dell-notebook*, *Apple-iPod*,

[3] Although there exist different types of term association relationships in WordNet [13] such as "Bush versus President of US" as synonym, or "G.W. Bush versus R. Reagan" as coordinate terms, these types of relationships are limited to addressing topical relationships.

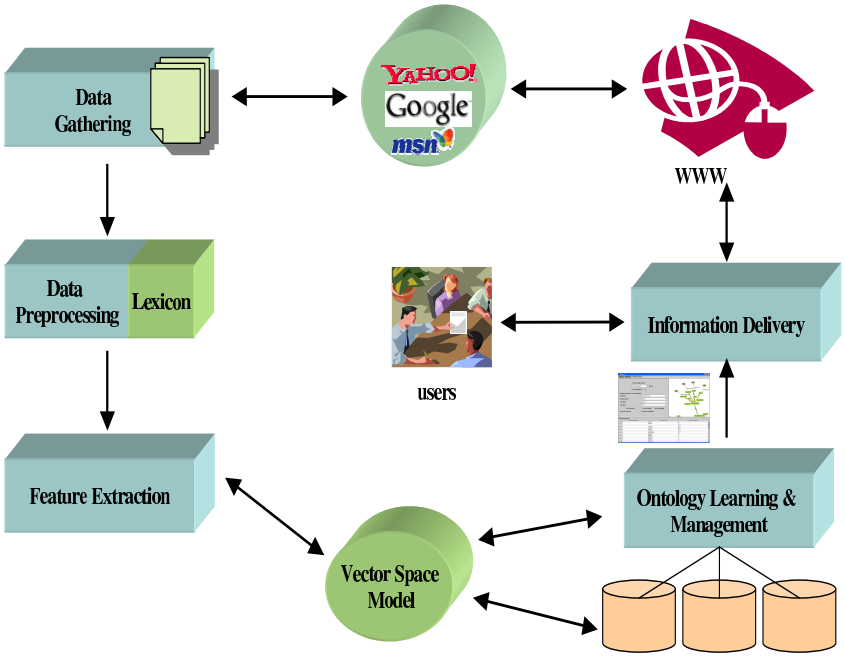

Fig. 1. Overview of a proposed framework

etc. Thus, concept-based ontologies have a limitation in supporting a topical search. To address this problem, we also demonstrate how our WebSim model can generate topical relations. In sum, the purpose of this research is to move one step forward to achieving the development of a novel feature extraction and similarity model that can be utilized for any ontology learning framework.

The remainder of this paper is structured as follows. In Section 2, we briefly review the related work, and highlight the strengths and weaknesses of the previous work in comparison with ours. In Section 3, we explain our feature extraction algorithm. Section 4 explores how to compute similarity between terms based on the extracted features. Section 5 explains how to identify key candidate terms that should be used for ontology modification. In Section 6, we present a method on how to relate WebSim and existing ontologies. Finally, we conclude the paper and provide our future plan in Section 7.

2 Related Work

Computation of the similarity between terms is at the core of ontology learning problem. There have been many attempts on automatic detection of similar words from text corpora. One of the widely used approaches in similarity computation is based on distributional hypothesis [4, 18]. That is, if words occur in a similar context, then they tend to have similar meanings. The context can

Notation	Meaning
t_i	An i-th term
D_i	A set of Web pages (returned by a search engine) for t_i
f_{ij}	A j-th feature of a term t_i
d_k	A k-th document returned by a search engine
l_k	The document length of d_k
$freq_{ijk}$	Term frequency of a feature f_{ij} in d_k
tf_{ijk}	Normalized term frequency of a feature f_{ij} in d_k
N_i	The size of D_i
n_{ij}	The number of documents in D_i where f_{ij} occurs at least once
w_{ij}	The weight of f_{ij}
$IC(t_i)$	The information content of t_i
$p(t_i)$	The concept probability of how much t_i occurs
$count(t_i)$	The term frequency of t_i on corpus
$concept\_freq(t_i)$	The concept frequency of t_i on corpus
n	The total number of terms observed in corpus

Table 1. Notations for WebSim

be defined in diverse ways. For example, it can be represented by co-occurrence of words within grammatical relationships (e.g., a set of verbs which take the word as a subject or object, a set of adjectives which modify the word, etc), or co-occurring words within a certain length of a window. Each context is referred to as features of a term. Thus, the set of all features of a term t_i constitutes the feature vector of t_i.

Recently, there have been research efforts for building ontologies automatically. In particular, text mining tools have been widely used to build ontologies [14, 11, 15, 5, 22]. In order to obtain context, current text mining research usually utilizes the Web as corpus. That is, a Web crawler is used to download certain amount of Web pages. Each Web page is then preprocessed by removing HTML tags, performing tokenization, etc. In addition, in order to obtain robust statistics, the process of eliminating duplicate documents may be conducted. Based on that, a feature extraction process is performed to identify key features for each term.

Our approach is different from the previous work in that we utilize a Web search engine to exploit the full content of the Web. That is, rather than relying on the snapshot of the Web, we can access as much content as possible (depending on how much content a search engine spider can crawl). Thus, our method is less sensitive to the problem of data sparseness. In addition, our feature extraction methodology is different from other approaches in that the context of terms are defined by a set of highly relevant documents returned by a search engine. Note that our research is complementary to the previous ontology learning efforts in that the extracted features or term similarity can be utilized for any ontology learning framework.

3 Feature Extraction for WebSim

In this section, we discuss how to extract features for each term. We first clarify between"term" and "word". Although "term" and "word" have same meanings in general, for the notational convention, "term" will be used to refer to the entity of similarity computation (i.e., we measure similarity between terms) while "word" is used to represent the feature of "term". Thus, a term is represented by a set of words, where each word corresponds to the feature of the term. Table 1 also illustrates the notations that will be used throughout this paper.

Extracting meaningful features for WebSim consists of three phases: retrieval of Web documents for each term, preprocessing of the retrieved documents, and construction of a vector space model with a relevant feature extraction method.

A Web search engine is necessary to obtain the initial set of relevant documents for each term. Toward this end, we use the open source software, Google Web API [23], which allows us to query billions of web pages directly in our own programs [4].

In the preprocessing step, meaningful information are extracted from Web pages using standard IR tools as follows:

- *HTML preprocessing.* Since downloaded pages are in HTML format, we remove irrelevant HTML tags, and extract meaningful (e.g., Javascript code is removed) information.
- *Tokenization.* Its main task is to identify the boundaries of the words.
- *Stemming.* There can be different forms for the same words (e.g., *students* and *student*, *go* and *went*). We need to convert these different forms of the same word to their root. Toward this end, instead of solely relying on Porter stemmer [19], in order to deal with irregular plural/tense, we combine Porter stemmer with the lexical database [12].
- *Stopwords removal.* Stopwords are the words that occur frequently in text but do not carry useful information. For example, *have*, *did*, and *get* are not meaningful. Removing such stopwords provides us with a dimensionality reduction effect. Toward this end, we employ stopword list used in Smart project [21]. Web-specific stopwords (e.g., host, click) are also added to our list.

After preprocessing, a term (t_i) is represented as a vector in a vector space [21]. The simple way to do this is to employ *bag-of-words* approach. That is, all content-bearing words in each document are taken and any structure of text or the word sequence is ignored. We treat each word as a feature of t_i, and represent each term as a vector of certain weighted word frequencies in this feature space. The weight of a word for each term is determined based on the following two heuristics.

[4] Since the Google API can use different data centers, the results returned by the API may be different from what we obtain by querying directly to "www.google.com" in a browser.

– Important words occur more frequently within a document than unimportant words do.
– The more times a word occurs throughout the documents within D_i, the stronger its predicting power becomes.

The term frequency (TF) is based on the first heuristic[5]. In addition TF can be normalized to reflect different document length. Let f_{ij} be the j-th feature of t_i, and $freq_{ijk}$ be the number of f_{ij}'s occurrences in a document d_k where $d_k \in D_i$. Then, term frequency (tf_{ijk}) of f_{ij} in d_k is defined as follows:

$$tf_{ijk} = \frac{freq_{ijk}}{l_k} \tag{1}$$

where l_k is the length of d_k.

The second heuristics is related with the document frequency (DF) of the word (the percentage of the documents that contains this word). In traditional IR research, inverse document frequency (IDF) has been widely used based on the observation that low document frequency words tend to be particularly important in identifying relevant documents with respect to a query. That is, words with high document frequency tend to occur in many irrelevant documents because the number of relevant documents to a query is generally small. However, in WebSim, since only relevant documents with respect to a term are retrieved, and stopwords are removed in the preprocessing step, a word with high document frequency within D_i is considered to be of a particular relevant feature for a term.

A combination of TF and DF introduces a new ranking scheme, which is defined as follows:

$$w_{ij} = \frac{n_{ij}}{N_i} \times \frac{\Sigma_{d_k \in D_i} tf_{ijk}}{N_i} \tag{2}$$

where w_{ij} is an weight of f_{ij}, N_i is the total number of documents in D_i, and n_{ij} is the number of documents in D_i where f_{ij} occurs at least once.

By exploiting the fact that only the top (high-weighted) few features contribute substantially to the norm of the term, we only keep the high-weighted features that make up most of the norm (80% or so). This approach reduces the number of features significantly while it minimizes the loss of information.

Table 2 shows the sample features for some terms[6]. As shown, the top features for each term characterize descriptive concepts of terms. For example, consider "RDF" and "DAML", which are Semantic Web languages. As expected, key concepts that describe RDF and DAML are extracted as features. Note that the extracted features sometimes do not always correspond to definitions of terms. For example, for a term "knowledge discovery", "sigkdd" is not a feature for defining the term. However, this is an important feature in that it is one of the largest organization in data mining. Therefore, extracted features by WebSim reflect the current trend on the term besides definition for the term.

[5] In WebSim, term frequency of t_i is counted in a document in D_i where D_i is referred to as a set of top most relevant documents for t_i (returned by a search engine).

[6] Those terms are selected from WebLearn [10].

Term	Features
Semantic web	web semant rdf xml data work inform uri languag document w3c technolog group resourc schema comput applic publish
Ontology	ontolog class properti rdf owl knowledg term definit inform formal agent descript semant defin languag daml logic oil
DAML	daml rdf ontolog languag oil web semant class org servic properti xml list inform resourc w3 releas markup id
RDF	rdf org xml resourc web document w3c rss list syntax descript properti inform w3 metadata schema version class uri graph
metadata	metadata inform standard resourc xml schema document data librari web meta descript core dublin element digit w3c rdf
XML	xml web develop document work data rss servic languag group java inform feed format resourc w3c dtd markup standard xsl
Bioinformatics	bioinformat biotech comput biologi genom protein scienc search journal sequenc biolog inform databas molecular
Computational biology	comput biologi research bioinformat journal genom inform scienc center databas molecular analysi search life sequenc
Gene expressions	gene express research genet human molecular biologi genom link develop articl project evolut cell protein journal time
Gene Ontology	gene ontolog databas link term protein annot search data tool consortium product genom function biolog molecular
Proteins	protein structur acid amino fold web databas search inform sequenc function link domain chain genom releas molecul cell
Parallel Programming	parallel program number mpi comput code cs perform applic openmp algorithm implement memori access scienc processor
Computer architecture	comput architectur memori paper book system perform parallel instruct project scienc design circuit high pipelin
Genetic algorithms	genet algorithm comput ai program learn evolutionari artifici ga guid research intellig inform problem network
Fitness function	function fit genet algorithm problem select gener ga solut comput search optim inform object program
Perceptron	perceptron learn input network train weight neuron output function connect neural pattern singl vector
Self organizing maps	map organ data som neural kohonen network learn vector wsom model visual cluster text similar inform
Knowledge discovery	data mine knowledg discoveri kdd number confer sigkdd acm research inform volum web search scienc
Web Usage Mining	web mine usag data user pattern public discoveri log research analysi inform project access server transact
Text mining	text mine data inform research search web extract document softwar comput analysi intellig retriev

Table 2. Sample features for terms. Note that all features are stemmed. For example, "inform" refers to "information", and so on.

t_i	t_j	$Sim_1(t_i, t_j)$
Data mining	Knowledge discovery	0.778
Data mining	Sequential patterns	0.422
Data warehouses	Knowledge discovery	0.510
Text mining	Information extraction	0.435
Computer vision	Image understanding	0.521
Genetic programming	Evolutionary computation	0.469
Encryption	WWW security	0.554
Natural language processing	NLP	0.591
Genetics	Gene expressions	0.596
Genetics	Human genome database design	0.689

Table 3. Sample term pairs that have high $Sim_1(t_i, t_j)$

4 A Step forward to Dynamic Semantics

Once each term is represented as a vector in a feature space, the next step is to measure closeness between two terms. Toward this end, we employ a Cosine metric which has been widely used in much information retrieval literature [21]. It measures similarity of two items according to the angle between them. Thus, vectors pointing to similar directions are considered as representing similar concepts. The cosine of the angle between two vectors t_i and t_j is defined by

$$Sim_1(t_i, t_j) = Cosine(v_i, v_j) = \frac{\sum_{k=1}^{n} w_{ik} \cdot w_{jk}}{||t_i|| \cdot ||t_j||} \tag{3}$$

where v_i and v_j correspond to the vectors of t_i and t_j, respectively.

The underlying assumption of the proposed approach is simple but effective: if a term t_i (e.g., ipod) and t_j (e.g., Apple) have some relationships, then the Web pages returned by t_i and t_j would be somewhat similar, consequently, similarity between v_i and v_j have high similarity (by Cosine metric). Table 3 illustrates this. This is generally true if a term is specific or non-ambiguous. However, this does not always hold due to the following reason.

One of the challenging problems in WebSim is how to deal with ambiguity of terms. That is, if a term has multiple meanings, then the dispatched Web pages are un-correlated to each other. For example, "clustering" has two meanings in top 10 ranked pages returned by Google (data mining and computer architecture context). Thus, even though we expect high similarity between "clustering" and "data mining", due to the ambiguity of "clustering", actual similarity between two terms becomes low. This problem is also inherent in a Web search. User queries tend to be short in general, consequently, they may be ambiguous, which often leads to irrelevant results.

Table 4 shows top high-weighted features for six ambiguous terms. For example, consider "classification". Due to the generality of this term, only one page is

Term	Features
Clustering	cluster server softwar data technolog linux base inform window high applic search servic product load analysi releas featur group
Classification	classif link search onlin inform extern econom literatur number org list north web bookmark journal
OIL	oil energi industri shell locat ga chang product price servic bp drill origin compani petroleum
Mutation	mutat research issu volum databas journal copi page science direct elsevi regist login
Crossover	crossov network web offic capabl linux custom softwar secur microsoft profession featur copi window bui
Selection	select search inform web servic internet scienc map natur access public onlin journal research technolog human

Table 4. Features for ambiguous terms

related with "classification" in data mining context. Moreover, for a term "oil", all extracted features are related with gasoline because top 50 pages returned by Google are all pointers to information on gas oil[7]. These are useful features if a knowledge engineer wants to add "oil" (in terms of energy context) into ontologies. However, if the knowledge engineer considers OIL (Ontology Inference Layer) in Semantic Web context, then the extracted features are problematic. Assuming "oil" in an energy sense is already in ontologies (because it was coined a long time ago), OIL in a Semantic Web sense is of particular interest in terms of enriching ontologies (because it is neology). Furthermore, consider "selection", "crossover" and "mutation" that are three main operators in genetic algorithms. Due to the generality of terms, extracted features do not represent distinctive characteristics of genetic algorithms.

In previous information retrieval research, query expansion has been widely studied in order to provide more useful search results in terms of refining a query by adding additional relevant search terms. The key point here is that the added terms should be somewhat related with the original query term. Otherwise, query expansion leads to a degradation of precision [6].

Suppose t_i and t_j are in consideration of similarity computation. If the similarity between t_i and t_j is not high enough, then combined queries (i.e., t_it_j and t_jt_i) are issued as queries to a Web search engine, and top N documents for both terms are retrieved. As discussed, if t_i and t_j are not related with each other, then adding an additional term will not be helpful, consequently, similarity between t_i and t_jt_i (and between t_j and t_it_j) will be still not high. However, if t_i and t_j are related with each other, then expanding t_i with t_j will result in high

[7] This is because Web search engines tend to rank a page based on how a query is matched with the page (i.e., whether the query is matched with title, etc) and its popularity using the notion of authorities and hubs [7, 1].

Term	Features
Classification library	librariclassif book congress subject scienc inform histori languag dewei list class author onlin catalog document
Classification clustering	cluster classif data class analysi method algorithm text inform distanc group list network imag fuzzi vector type similar variabl model hierarch program point document
Clustering architecture	cluster architectur manag server applic network databas servic group avail softwar replic microsoft
Clustering mining	data mine cluster algorithm group model web inform custom product text learn knowledg method similar
Mining gold	gold mine miner inform histori pan prospect compani south stock state rock rush corpor metal resourc california
Mining data	data mine statist analysi inform softwar web applic model knowledg product book databas process research intellig
Linux automata	linux *automata* program softwar version *cellular* simul org game *life* file comput window java *state model* 3d

Table 5. Sample features for terms with context

similarity (i.e., similarity between t_i and $t_j t_i$ is expected to be high). In this step, both $t_i t_j$ and $t_j t_i$ are submitted to the Web search engine. The reason is that the order of query terms affects the search results. Thus, when the similarity between t_i and t_j is not high enough, the similarity will be refined as follows:

$$Sim_2(t_i, t_j) = Average(Cosine(t_i, t_j\ t_i),\ Cosine(t_j, t_i\ t_j)) \tag{4}$$

Table 5 shows how term expansion can be used to narrow down the sense of a term. For example, since "classification" and "clustering" are related in data mining context, adding "clustering" to "classification" will refine the meaning of "classification". This is because there exist a sense that both terms share even though they have multiple meanings. As a result, extracted features on "classification clustering" are on data mining subject. However, expanding "linux" with "automata" destroys the true characteristics of the term rather than refines the meaning. Consequently, resulted features are distorted (distorted features are shown in italic). Therefore, term expansion is helpful only when a relevant term is added.

In general, the following characteristics hold for our similarity model.

- Characteristic 1: $Sim_k(t_i, t_i) = 1$ for $k = 1, 2$.
- Characteristic 2: $Sim_k(t_i, t_j) = Sim_k(t_j, t_i)$ for $k = 1, 2$.
- Characteristic 3: $Sim_1(t_i, t_j\ t_i) \geq Sim_1(t_j, t_i\ t_j)$ if t_i is a more specific (or less ambiguous) term than t_j.

t_i	t_j	$Sim_1(t_i, t_j t_i)$	$Sim_1(t_j, t_i\ t_j)$
Laptop	Notebook	0.895	0.153
Data Mining	Classification	0.838	0.227
Data Mining	Clustering	0.769	0.521
RDF	OIL	0.845	0.060
Yahoo	Messenger	0.690	0.642

Table 6. Illustration of the third characteristic of WebSim

The first two characteristics are obvious in that similarity relations are reflexive and symmetric in general. The third one indicates that the similarity between a specific term (t_i) and a general term (t_j) with refinement has a better chance of getting higher similarity than the similarity between a general term and a specific term with refinement. Table 6 illustrates this characteristic. For example, "notebook" has two meanings (movie title and computer) in the Web while "laptop" has only one meaning. Thus, refinement of "notebook" with "laptop" transforms the meaning of "notebook laptop" as a computer sense. Thus, similarity between "notebook laptop" and "laptop" is very high.

5 Candidate Term Derivation for Ontology Modification

One of the key issues in ontology enrichment is how to identify candidate terms that should be added into an ontology. In the following, we describe two approaches.

1. In our previous research efforts, we presented topic mining, which effectively identifies useful patterns (e.g., news topics or events, key terms at multiple levels of abstraction) from news streams [3, 2]. The proposed framework is unique in that the key topical terms are dynamically generated based on incremental hierarchical document clustering on news streams. The presented clustering algorithm has several key advantages, including the scalability with the high dimensionality, ability to generate a cluster hierarchy dynamically, capability to discover clusters with different shapes and sizes, and ability to provide succinct description of clusters. The topic mining framework can complement to our WebSim in that it can automatically identify key candidate terms/concepts from Web document streams. The identified candidates can be given as input to our WebSim for the purpose of ontology enrichment.
2. The feature extraction methodology (in Section 3) can be effectively used for candidate term generation. That is, as shown in Table 2, since features for each term are somewhat related with the term, a term in the ontology can be submitted as a query to a Web search engine. The obtained features, which do not exist in the ontology, can be candidates for ontology enrichment.

Besides adding a new term into an existing ontology, the ontology should be restructured (i.e., existing term relationships in ontologies can be changed) as time evolves. Toward this end, we present WebSim-based approach to select candidate term pairs that should be modified. Due to the large number of terms in ontologies, it is computationally expensive to compute pairwise similarity between all terms in the ontology (this is because we cannot predict which part of the ontology should be modified, e.g., consider OIL and XML that are far from each other in conventional conceptual ontologies). That is, if the size of the ontology is m, then we need $O(m^2)$ similarity computations. However, if we utilize WebSim, then the number of computations can be drastically reduced. As discussed, for each term, extracted features by WebSim represent the up-to-date knowledge on the term. Thus, rather than examining all term pairs in the ontology, for each term (t_i), we only need to compute similarity between t_i and the features of t_i that are in the ontology. Assuming that the number of the features for each term is constant (because only top weighted features are considered), the complexity can be reduced to $O(m)$ from $O(m^2)$, which is a significant improvement.

6 Relations between WebSim and Semantic Similarity

In this section, we present a methodology on how to investigate relatedness between WebSim and existing ontologies like WordNet.

Given a pair of terms, t_i and t_j, a simple similarity measure in ontologies is to use the edge counting method where the distance corresponds to the number of edges between terms in the ontology. The shortest or the average distance can be used if there exist multiple paths. Thus, the shorter the path from one term to another, the more similar they are. However, this approach relies on the assumption that links in the taxonomy represent uniform distances, which does not hold in many existing ontologies. Hence, it cannot provide correct similarity estimation in general.

Recently, alternative methods have been proposed to evaluate semantic similarity in a taxonomy based on information content [9, 20]. These approaches rely on the incorporation of empirical probability estimates into a taxonomic structure. Previous study has shown that this type of approaches is significantly less sensitive to link density variability.

The information content of a term t_i ($IC(t_i)$) can be quantified as follows:

$$IC(t_i) = -log(p(t_i)) \tag{5}$$

where $p(t_i)$ is the probability of how much a term t_i occurs. Frequencies of terms can be estimated by counting the number of occurrences in corpus. Each term that occurs in the corpus is counted as an occurrence of each concept containing it.

$$concept\_freq(t_i) = \sum_{t_j \in C_{t_i}} count(t_j) \tag{6}$$

where C_{t_i} is the set of terms subsumed by a term t_i. Then, concept probability for t_i can be defined as follows:

$$p(t_i) = \frac{concept\_freq(t_i)}{n} \tag{7}$$

where n is the total number of terms observed in corpus.

Equation (5) states that informativeness decreases as concept probability increases. Thus, the more abstract a concept, the lower its information content. This quantization of information provides a new approach to measure semantic similarity. The more information two terms share, the more similar they are. Resnik [20] defines the information shared by two terms as the maximum information content of the common parents of the terms in the ontology (Equation (8)).

$$Resnik(t_i, t_j) = max_{t \in CP(t_i, t_j)}[-log(p(t))] \tag{8}$$

where $CP(t_i, t_j)$ represents the set of parents terms shared by t_i and t_j.

Because the value of Equation (8) can vary between 0 to infinity, we use Lin's metric instead [9], which is defined as follows:

$$Lin(t_i, t_j) = \frac{2 \times max_{t \in CP(t_i, t_j)}[-log(p(t))]}{(IC(t_i) + IC(t_j))} \tag{9}$$

The Equation (9) varies between 0 (dissimilarity) and 1 (similarity).

Table 7 shows semantic similarity and WebSim of selected terms. Semantic similarity is obtained by using WordNet::Similarity [17]. As expected, due to the lack of ability to express topical relations in WordNet, we observed low semantic similarity for the first five term pairs. In contrast, our WebSim model successfully captured the similarity relations. Even though $Sim_1(t_i, t_j)$ was low for "automata" and "Turing machine", this is because of the ambiguity of "automata", which has two meanings (in theory of computation and computational learning context). For the term pairs that were detected by semantic similarity very well (such as "nurse" vs "doctor"), our WebSim could also identify high similarity using refinement. Finally, for the terms that do not exist in WordNet (e.g., Ipod or Microsoft), WebSim could capture high similarity. In sum, WebSim performs well on high semantic similarity term pairs using refinement while uncovers topical relations that do not exist in WordNet.

7 Conclusion and Future Works

We presented a feature extraction and similarity framework, referred to as WebSim, which is vital to intelligent information retrieval. In order to accommodate dynamically changing knowledge, we developed the similarity model based on a Web Search engine. In addition, to identify candidate terms for ontology modifications, we presented two methods that can be effectively utilized for any ontology learning framework. Finally, coupling with semantic similarity, we demonstrated how WebSim could identify unknown relations in WordNet.

t_i	t_j	$Lin(t_i, t_j)$	$Sim_1(t_i, t_j)$	$Sim_1(t_i, t_j t_i)$	$Sim_1(t_j, t_i t_j)$
Semantics	Metadata	0	0.171	0.566	0.741
Firewall	Encryption	0	0.218	0.768	0.631
Automata	Turing machine	0	0.078	0.248	0.865
Apple	Computer	0.121	0.153	0.660	0.341
Yahoo	Messenger	0.230	0.102	0.690	0.642
Tomato	Vegetable	0.853	0.127	0.890	0.591
Doctor	Nurse	0.797	0.091	0.481	0.791
Microsoft	Windows	Undef	0.541	0.772	0.794
Apple	Ipod	Undef	0.644	0.771	0.982

Table 7. WebSim versus semantic similarity. Undef denotes that the term does not exist in WordNet.

We intend to extend this work in terms of more sophisticated feature weighting. Based on the observation that the Web page with high rank is generally more informative than the Web pages with low rank, each Web page can be weighted by order when we extract features. That is, the features in the first page should be weighted higher than the features in the 20-th page, and so on. Therefore, it is worthwhile to investigate how feature weighting by rank affects the similarity model.

8 Acknowledgement

This research has been funded in part by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152.

References

1. S. Brin, and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference*, 1998.
2. S. Chung, and D. McLeod. Dynamic topic mining from news stream data. In *Proceedings of the 2nd International Conference on Ontologies, Databases, and Application of Semantics for Large Scale Information Systems*, 2003.
3. S. Chung, and D. McLeod. Dynamic pattern mining: an incremental data clustering approach. *Journal on Data Semantics*, 2:85-112, 2005.
4. I. Dagan, F. Pereira, and L. Lee. Similarity-based estimation of word cooccurrence probabilities. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 1994.
5. E.J. Glover, D.M. Pennock, S. Lawrence, and R. Krovetz. Inferring hierarchical descriptions. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management*, 2002.

6. L. Khan, D. McLeod, and E.H. Hovy. Retrieval effectiveness of an ontology-based model for information selection. *The VLDB Journal*, 13(1):71-85, 2004.
7. J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 1998.
8. D. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd. Cyc: Toward programs with common sense. *Communications of the ACM*, 33(8):30-49, 1990.
9. D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.
10. B, Liu, C.W. Chin, and H.T Ng Mining topic-specific concepts and definitions on the web. In *Proceedings of the 12th International World Wide Web Conference*, 2003.
11. A. Maedche, and S. Staab. Ontology learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2), 2001.
12. I.D. Melamed. Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons. In *Proceedings of the 3rd Workshop on Very Large Corpora*, 1995.
13. G. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235-312, 1990.
14. M. Missikoff, P. Velardi, and P. Fabriani. Text mining techniques to automatically enrich a domain ontology. *Applied Intelligence*, 18(3):323-340, 2003.
15. J. Nemrava, and V. Svátek. Text mining tool for ontology engineering based on use of product taxonomy and web directory. In *Proceedings of the Dateso Annual International Workshop on DAtabases, TExts, Specifications and Objects*, 2005.
16. N.F. Noy, M. Sintek, S. Decker, M. Crubézy, R.W. Fergerson, and M.A. Musen. Creating and acquiring Semantic Web contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60-71, 2001.
17. T. Pedersen, S. Patwardhan, and J. Michelizzi. WordNet::Similarity - measuring the relatedness of concepts In *Proceedings of the 5th Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, 2004.
18. F. Pereira, N.Z. Tishby, and L. Lee. Distributional clustering of english words. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, 1993.
19. M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130-137, 1980.
20. P. Resnik. Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 1999.
21. G. Salton and M.J. McGill. *Introduction to modern information retrieval.* McGraw-Hill, 1983.
22. M. Sanderson, and W.B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.
23. Google Web APIs. http://www.google.com/apis/.