

WebSim: A Novel Term Similarity Metric based on a Web Search Technology

Seokkyung Chung^{1*}, Jongeun Jun^{2**}, and Dennis McLeod²

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. Given that pairwise similarity computations are essential in ontology learning and data mining, we propose WebSim (Web-based term Similarity metric), whose feature extraction and similarity model is based on a conventional Web search engine. There are two main aspects that we can benefit from utilizing a Web search engine. First, we can obtain the freshest content for each term that represents the up-to-date knowledge on the term. This is particularly useful for dynamic ontology management in that ontologies must evolve with time as new concepts or terms appear. Second, in comparison with the approaches that use the certain amount of crawled Web documents as corpus, our method is less sensitive to the problem of data sparseness because we access as much content as possible using a search engine. At the core of WebSim, we present two different methodologies for similarity computation, a mutual information based metric and a feature-based metric. Moreover, we show how WebSim can be utilized for modifying existing ontologies. Finally, we 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

With the rapid growth of the World Wide Web, Internet users are now experiencing overwhelming quantities of online information. Since manually analyzing the data becomes nearly impossible, the analysis would be performed by intelligent information management techniques to fulfill users' information needs quickly.

Representation and extraction of semantic meanings from information contents is essential in intelligent information management. This issue has been explored in diverse research disciplines including artificial intelligence, data mining, information retrieval, natural language processing, etc. One of the widely used approaches to addressing this problem is to exploit ontologies. For example, when users use irrelevant keywords (due to their broad and vague information

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

** To whom correspondence should be addressed.

needs or unfamiliarity with the domain of interests), query expansion based on ontologies can improve retrieval accuracy by providing an intelligent information selection.

A knowledge acquisition problem (i.e., how to build ontologies) is one of the main bottlenecks in ontology-based approaches. Although there exist hand-crafted ontologies such as WordNet [13] or CYC [9], significant amounts of domain-specific terms (e.g., scientific or engineering terms) or neology are not present in general-purpose ontologies. Thus, it is essential to build ontologies that can characterize given applications.

However, although ontology-authoring tools have been developed in the past decades [17, 25], constructing ontologies by hand whenever new domains are encountered needs significant amount of time and efforts. Additionally, since ontologies must evolve with time as new concepts or terms appear, it is essential to maintain existing ontologies up-to-date. Therefore, ontology learning, which is a process of integrating knowledge acquisition with data mining, becomes a must. Consequently, a knowledge expert can build and maintain domain ontologies more efficiently with the support of ontology learning. Given that computation of the similarity between terms is at the core of ontology³ learning problems, we focus our attentions on computing similarity between terms.

As the Web continues to grow as a vehicle for the distribution of information, the massive 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. The proposed approach takes advantage of two main aspects of the Web search engine technology. First, as many thousands of Web pages are published daily on the Web, the Web reflects and characterizes current trend of knowledge. Thus, we can obtain the freshest content for each term that represents the up-to-date knowledge on the term. This is particularly useful for dynamic ontology management in that ontologies must evolve with time as new concepts or terms appear. Second, because we access as much content as possible using search engines, our method is less sensitive to the problem of data sparseness. 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.

At the core of WebSim, we present two similarity metrics, a frequency-based metric and a feature-based metric. The frequency-based one is a mutual information based measure that utilizes the number of Web pages associating with each term. In contrast, the feature-based similarity metric extracts relevant features for each term, and performs similarity computation based on the extracted features. With the feature-based metric, we present how to deal with ambiguous terms for the similarity computation, which is one of the difficult problems in a

³ Our definition of ontologies is a collection of key concepts and terms along with their inter-relationships. Although ontologies are often equipped with a set of inference rules and constraints that are used to reason about concepts, this notion is beyond the scope of this paper.

Web search. We also show how ontologies can be restructured or enriched with WebSim. Finally, we demonstrate the characteristics of WebSim by coupling with WordNet.

One of the main problems in concept-based ontologies is that topically related concepts and terms are not explicitly linked. That is, there is no relation between *Dell-notebook*, *Apple-iPod*, etc. 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. Thus, concept-based ontologies have a limitation in supporting a topical search. For example, consider the Sports domain ontology that we have developed in our previous work [7]. In this ontology, “Kobe Bryant”, who is an NBA basketball player, is related with terms/concepts in Sports domain. However, for the purpose of query expansion, “Kobe Bryant” also needs to be connected with a “court trial” concept if a user keeps “Kobe Bryant court trial” in mind. Therefore, it is essential to provide explicit links between topically related concepts/terms. To address this problem, we also demonstrate how topical relations are generated in WebSim, and compare WebSim and semantic similarity in WordNet. 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 present an information-theoretic similarity measure. In Section 4, we explain our feature extraction algorithm and explore how to compute similarity between terms based on the extracted features. Section 5 explains how WebSim can be utilized for ontology modification. In Section 6, we discuss the characteristics of WebSim by relating general-purpose 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 [5, 19]. That is, if words occur in a similar context, then they tend to have similar meanings. The context can 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, 24, 16, 6, 23]. In order to obtain context, current text mining research usually utilizes the Web as corpus. 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.

Besides ontology learning, WebSim can be extended into other applications as well. For example, ontology matching, which aims at identifying mappings between related entities of multiple ontologies, has been widely studied recently [2, 26]. Many different matching solutions proposed so far exploit the various properties of data such as structures of ontologies, data instances, and string similarity using edit distance. WebSim is expected to reinforce previous matching technologies in that it is a similarity metric that is orthogonal to existing ones. Moreover, with the popularity of online sellers' product recommendation to their customers based on purchasing patterns, ontologies can be utilized to customizing a system to a user's preferences in e-commerce [27]. That is, ontologies can be effectively used for modelling customers' behavior and users' profiles. WebSim is particularly useful in this type of application in that it can enrich existing taxonomy maintained by online shopping malls (e.g., amazon.com). In sum, WebSim is expected to be a key enabling technology for the tasks where pairwise similarity computations play a central role.

3 WebSim: A Simple Mutual Information Approach

In this Section, we present the first similarity metric, which is simple but powerful. The measure is referred to as an MI-based (Mutual Information) WebSim. Table 1 illustrates the notations that will be used throughout this paper.

The underlying assumption behind the MI-based WebSim is that two terms co-occur frequently if they are similar to each other. Mutual information is an information-theoretic metric that quantifies *relatedness* between two words. The mutual information between t_i and t_j is defined as follows:

$$MI(t_i, t_j) = \log \frac{p(t_i, t_j)}{p(t_i) \times p(t_j)} \quad (1)$$

The higher value in $MI(t_i, t_j)$ implies the stronger association between t_i and t_j .

Notation	Meaning
t_i	An i -th term
$df(t_i)$	A total number of Web pages that are matched with a query t_i
N	A total number of Web pages that a search engine indexes
$p(t_i)$	The probability that a search engine returns results for t_i
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
$prob(t_i)$	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
C	The size of corpus

Table 1. Notations for WebSim

Mutual information has been widely used in previous text mining research as a criteria for measuring term association. The probability is usually defined by term frequency divided by the total number of terms observed in corpus. However, this probability is restricted by size of corpus. In particular, if a term is a new coined one, then it suffers from a data sparseness problem. To address this problem, WebSim utilizes a search engine to estimate the probability approximately. Figure 1 illustrates the idea. A FE (Front-end) scraper sends a query to a Web search engine, and extracts the number of documents that a query is matched with⁴. In order to estimate the total number of documents a search engine indexes, based on the fact that most search engines supports a boolean query, the number of documents for two different queries (“ t_i ” and “NOT t_i ”) are summed up. Thus, WebSim defines $p(t_i)$ as follows:

$$p(t_i) = \frac{df(t_i)}{N} \quad (2)$$

where N is the total number of Web pages that a search engine indexes, and $df(t_i)$ is the total number of returned Web pages with respect to t_i . Consequently,

⁴ Although Google is used in this paper, because most search engines display the total number of documents that are matched with a query, WebSim can use other search engines as well. It is also worthwhile to investigate how different search engines affect WebSim, but out of scope in this paper.

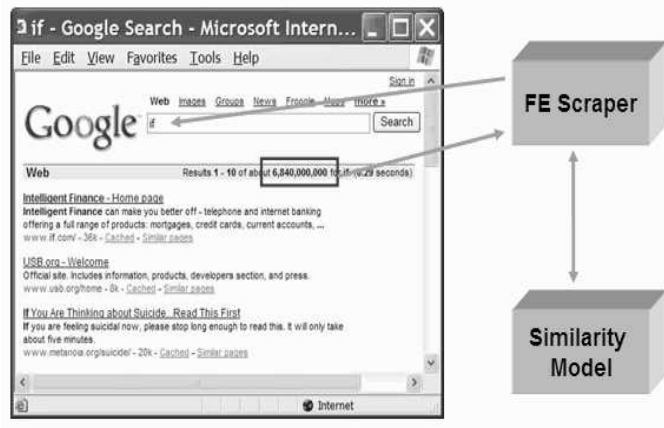


Fig. 1. Overview of an MI-based WebSim

in contrast to previous text mining approaches, WebSim uses the different notion of $p(t_i)$, which is the probability that a search engine returns results for t_i .

Most Web search engines provide advanced search features in that a user can specify how a query is matched with the page. That is, a user can retrieve the Web pages where a query is matched with title of the page (M_1), text of the page (M_2), URL of the page (M_3), links to the page (M_4), or anywhere in the page (M_5). Although it is worthwhile to investigate how different matching affects the accuracy of WebSim, this is beyond the scope of this paper. However, we briefly compare M_1 and M_5 .

Table 2 presents sample results. As shown, in most cases (Type 1), using both M_1 and M_5 , WebSim captures similarity between related term pairs fairly well. WebSim with M_1 is better than WebSim with M_5 in some cases (Type 2) such as *DAML-OIL*, or *notebook-computer*. Moreover, M_1 is able to adjust the similarity values that M_5 overestimates (*text mining-computational biology*). This is expected in that M_1 provides more accurate context for term co-occurrence than M_5 does. In some cases (Type 3), WebSim fails to capture similarity between term. This is primarily because mutual information prefers high-frequency terms to low-frequency ones (i.e., gives higher similarity values for low-frequency terms). For example, in case of *DAML-OIL* and *OWL-OIL*, because of relatively low frequency of “DAML” in comparison with “OWL”, WebSim captures association for *DAML-OIL* while it cannot for *OWL-OIL*. This also holds for *apple-computer*. Thus, although mutual information can detect pairwise similarity fairly well, besides relying on only $p(t_i)$, there is a need to incorporate content (associated with the term) into WebSim, which motivates the necessity of the second similarity metric.

Type	t_i	t_j	M_5	M_1
1	Natural Language Processing	NLP	7.71	7.31
	Self Organizing Maps	SOM	6.32	3.59
	Artificial Intelligence	AI	5.35	8.04
	Genetic Algorithm	Evolutionary Computation	8.60	8.47
	Data Mining	Clustering	4.18	4.06
	Data Mining	Knowledge Discovery	6.85	12.5
	Text Mining	Ontology	4.40	3.62
	Text Mining	ODBASE	5.33	7.12
	Text Mining	ODBASE	5.33	7.12
	Text Mining	Bioinformatics	4.87	4.19
	Computer Security	Firewalls	3.41	3.31
	Clustering	Classification	4.49	5.34
	Classification	Neural Networks	3.92	6.49
	Machine Learning	Text Mining	6.57	3.77
	Semantic Web	Ontology	5.39	7.33
	Semantic Web	DAML	5.86	4.87
	Bioinformatics	Computational Biology	5.47	9.96
	OWL	DAML	6.17	5.30
	ODBASE	Coopis	15.4	22.7
	ODBASE	DOA	10.0	14.6
	Neural Networks	Perceptron	8.35	6.84
	Neural Networks	Multi Layer Perceptron	8.92	10.4
	2	<i>DAML</i>	<i>OIL</i>	2.89
<i>Notebook</i>		<i>Computer</i>	0.05	4.38
Operating System		Unix	1.54	5.28
3	<i>Text Mining</i>	<i>Computational Biology</i>	3.70	-2.18
	OWL	metadata	0.71	-3.23
	<i>OWL</i>	<i>OIL</i>	-0.16	-3.98
	<i>Apple</i>	<i>Computer</i>	-0.90	0.28
	Data Mining	Classification	1.75	1.27

Table 2. Mutual information for sample term pairs

4 WebSim: A Similarity Model based on Feature Extraction

This section presents another similarity model based on feature extraction, which is referred to as a feature-based WebSim. Section 4.1 discusses feature extraction methodology. Section 4.2 explores similarity computation.

4.1 Feature Extraction

In this section, we explain 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.

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 [28]⁵.

In the preprocessing step, meaningful information are extracted from Web pages using standard IR tools. This process includes HTML preprocessing (e.g., removing irrelevant HTML tags or Javascript code, etc), tokenization, stemming with a Porter stemmer [20] and the lexical database [12], and stopwords removal [22]⁶.

After preprocessing, a term (t_i) is represented as a vector in a vector space [22]. The simple way to do this is to employ *bag-of-words* approach. 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.

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

The term frequency (TF) is based on the first heuristic. 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). 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:

$$tf_{ijk} = \frac{freq_{ijk}}{l_k} \quad (3)$$

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,

⁵ Alternatively, we can use the front-end scraper that is presented in Section 3.

⁶ Web-specific stopwords (e.g., host, click) are also added to our list.

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:

$$w_{ij} = \frac{n_{ij}}{N_i} \times \frac{\sum_{d_k \in D_i} t f_{ijk}}{N_i} \quad (4)$$

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 3 shows the features (with high weights) for sample terms. As shown, the top features for each term characterize descriptive concepts of terms. For example, consider “knowledge discovery” and “association rules”. As expected, key concepts that describe both terms 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 organizations in data mining. Similarly, “agraw” (Rakesh Agrawal), who is an inventor of association rule mining, is extracted as a feature for “association rules”. Therefore, extracted features by WebSim reflect the current trend on the term besides definition for the term.

4.2 A Similarity Model based on Extracted Features

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 that has been widely used in previous information retrieval literature [22]. 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

$$Sim_1(t_i, t_j) = Cosine(v_i, v_j) = \frac{\sum_{k=1}^n w_{ik} \cdot w_{jk}}{\|t_i\| \cdot \|t_j\|} \quad (5)$$

where v_i and v_j correspond to the vectors of t_i and t_j , respectively. $Cosine(v_i, v_j)$ ranges from 0 to 1.

The underlying assumption of the proposed approach is simple but effective: if 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 becomes high (by Cosine metric). Table 4 illustrates this. This is generally true if a term is specific or non-ambiguous. However, this does not always hold due to the ambiguity of terms.

Term	Features
Knowledge discovery	data mine knowledg discoveri kdd number confer sigkdd acm research inform volum web search scienc
Association rules	rule associ data item mine transact databas inform agraw algorithm confid analysi stream discoveri knowledg itemset
Sequential patterns	pattern mine sequenti data time sequenc databas stream algorithm associ agraw rule transact srikant frequent tempor
Decision trees	tree decis data inform node learn classific algorithm test split predict gain class train text machin model classifi attribut
Data warehouses	data warehous inform wareh busi manag softwar databas integr enterpris solut intellig servic server view decis
Object recognition	object recognit imag model vision base featur comput match visual scene view research geometr invari data shape recogn
Multi layer perceptron	layer perceptron multi network output function input neural weight learn hidden unit model neuron error linear mlp
Self organizing maps	map organ data som neural kohonen network learn vector wsom model visual cluster text similar inform
Parallel computer architecture	comput parallel architectur softwar hardwar design program memori multiprocessor perform morgan kaufmann network
Firewalls	firewal internet secur network comput connect protect softwar servic window filter packet inform server host port
Cryptography	cryptographi secur crypto inform kei privaci encrypt rsa archiv research cryptolog softwar algorithm pgp code
Machine translation	translat machin english languag french onlin text mt spanish comput german public chines associ research
Multimedia	multimedia inform video time grolieronlin photo media histori nation web archiv imag stori real flash audio view
Virtual reality	virtual realiti 3d world vr research model list time comput simul vrml applic interact commun environ motion view
Push down automata	automata push state context languag stack free pda formal grammar program correct inform input finit regular theori
Finite Automata	finit automata state algorithm determinist languag regular machin string dfa comput transit express nfa
Turing machines	machin ture comput state tape program number symbol halt run problem instruct left alan cell blank function
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

Table 3. Sample features for terms. Note that all features are stemmed. For example, “inform” refers to “information”, and so on.

t_i	t_j	$Sim_1(t_i, t_j)$
Semantic Web	XML	0.405
Genetic algorithms	Evolutionary computation	0.433
Encryption	Computer security	0.535
Information warfare	Computer security	0.444
Parallel computing	Computer architecture	0.623
Parallel programming	MPI	0.383
Data warehouses	Knowledge discovery	0.510
Data mining	Knowledge discovery	0.778
Natural language processing	Computational linguistics	0.439
Natural language processing	NLP	0.583

Table 4. Sample term pairs that have relatively high $Sim_1(t_i, t_j)$

One of the challenging problems in the feature-based WebSim is how to deal with ambiguity of terms. That is, if a term has multiple meanings, then the returned Web pages are somewhat 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). Consequently, due to the ambiguity of “clustering”, actual similarity between “clustering” and “data mining” becomes low even though clustering is one of the subfields in data mining. This problem is also inherent in a Web search. Because user queries usually tend to be short, they may be ambiguous, which often leads to irrelevant search results.

Table 5 shows top high-weighted features for six ambiguous terms. For example, consider “classification”. Due to the generality of this term, none of the top 10 ranked pages returned by Google is related with classification in data mining context. Moreover, regarding to “oil”, most top 50 pages returned by Google are all pointers to information on gas oil, and only 2 pages are about Ontology Inference Layer. Consequently, all extracted features for “oil” are related with gasoline. 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 [8, 1]. The extracted features for “oil” are useful ones if a domain expert wants to add “oil” (in terms of energy context) into ontologies. However, if he/she 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). Moreover, consider “selection”, “crossover” and “mutation” that are three main operators in genetic algorithms. Due to the generality of the 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. That is, a query can

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 5. Features for ambiguous terms

be refined 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 [7].

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., $t_i t_j$ and $t_j t_i$) 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 $t_j t_i$ (and between t_j and $t_i t_j$) 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 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 because 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:

$$Sim_2(t_i, t_j) = Average(Cosine(t_i, t_j t_i), Cosine(t_j, t_i t_j)) \quad (6)$$

Table 6 shows how term expansion can be used to refine the sense of a term. 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.

Term	Features
Oil odbase	ontolog oil web semant confer inform logic descript system odbase knowle languag gobl base proceed databas model
Agent coopis	agent system inform confer cooper univers comput coopi base paper web distribut knowledg data model servic
Agent insurance	insur agent life compani agenc state servic term licens inform financi busi brok onlin quot health nationwid auto
Classification clustering	cluster classif data class analysi method algorithm text inform distanc group list network imag fuzzzi vector type similar variabl model hierarch program point document
Clustering architecture	cluster architectur manag server applic network databas servic group avail softwar replic microsoft
Linux automata	linux <i>automata</i> program softwar version <i>cellular</i> simul org game <i>life</i> file comput window java <i>state model</i> 3d

Table 6. Sample features for terms with context

5 Ontology Modification with WebSim

Ontology modification is composed of two parts: ontology enrichment, and ontology restructuring. In this Section, we explore how to deal with this issue with WebSim. Figure 2 provides an overview of the proposed method.

One of the key issues in ontology enrichment is how to identify candidate terms that should be added into an ontology. In the past, 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 [4, 3]. Topic mining is a key enabling technology in ontology enrichment. The idea of coupling topic mining and ontology enrichment is as follows:

Topic mining sends a Web crawler to a collection of key sites that are related with the domain of interest (or collection of popular Web sites like CNN if we want to enrich general-purpose ontologies), and retrieves a set of domain specific documents. One of the main capabilities of topic mining is that the key topical terms are dynamically generated based on incremental hierarchical document clustering. Thus, the topic mining framework can complement to ontology enrichment in that it can automatically identify key candidate terms/concepts from Web document streams. The identified candidates can be given to a domain expert.

Moreover, the feature extraction methodology (in Section 4.1) can be effectively used for candidate term generation. That is, as shown in Table 3, since features for each term are key concepts that describe the main characteristics of the term, WebSim can use a term in the ontology to derive the features for the term. If the obtained features do not exist in the ontology, then domain experts can add the features for the purpose of ontology enrichment. Alternatively,

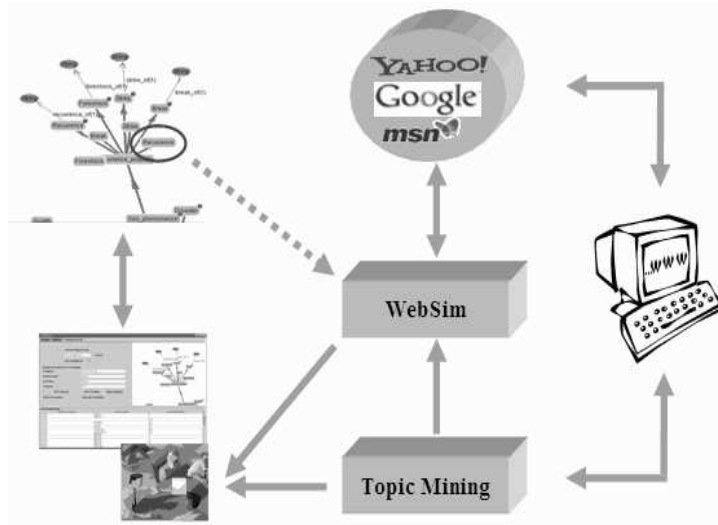


Fig. 2. Overview of the process for ontology modification with WebSim

terms identified by topic mining can be given as input to WebSim to populate candidate terms for ontology enrichment.

Besides adding a new term into an existing ontology, the ontology should be restructured as time evolves. That is, existing term relationships in ontologies need to be changed. Since it is extremely difficult to update ontology manually, it is necessary to suggest which parts of ontology have possibilities of modification. 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. Thus, if the size of the ontology is m , then we need $O(m^2)$ pairwise similarity computations. However, the number of computations can be significantly reduced if the WebSim is employed. 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.

It is worthwhile to compare the MI-based WebSim and the feature-based WebSim, so domain experts can choose the metrics for their own purpose. In terms of computational cost, the MI-based WebSim is cheaper than the feature-based WebSim. In our simulation, the difference between the number of docu-

ments of t_it_j and that of t_jt_i is 912.70. This number can be ignored considering that the average number of returned documents for the terms in our test is 8,267,363.81. Thus, for the MI-based WebSim, only 3 queries need to be submitted (i.e., t_i , t_j and t_it_j). In contrast, for the feature-based WebSim, both t_it_j and t_jt_i (as well as t_i and t_j) need to be submitted to a Web search engine because the order of query terms affects the top ranked Web pages, consequently, extracted features will be changed. In sum, the MI-based WebSim needs 3 query submissions while the feature-based WebSim needs 4 query submissions. Moreover, the feature-based WebSim needs the feature extraction step, and cosine similarity computations. Thus, the MI-based WebSim is computationally cheaper than the feature-based WebSim. However, the feature-based WebSim produces much more reliable similarity values than the MI-based WebSim does because mutual information is strongly influenced by the marginal probabilities of terms.

6 Semantic Similarity versus WebSim

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

Recently, semantic similarity metrics have been proposed to evaluate similarity between two terms in a taxonomy based on information content [10, 21]. 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:

$$IC(t_i) = -\log(p(t_i)) \quad (7)$$

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.

$$concept_freq(t_i) = \sum_{t_j \in C_{t_i}} count(t_j) \quad (8)$$

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:

$$prob(t_i) = \frac{concept_freq(t_i)}{C} \quad (9)$$

where C is the size of corpus, which corresponds to the total number of terms observed in corpus.

Equation (7) 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

t_i	t_j	Lin	MI	Sim_1	Sim_2
Semantics	Metadata	0	2.222	0.171	0.653
Firewall	Encryption	0	4.566	0.218	0.699
Automata	Turing machine	0	6.640	0.078	0.500
Apple	Computer	0.121	0.909	0.153	0.556
Yahoo	Messenger	0.230	3.795	0.102	0.666
Tomato	Vegetable	0.853	6.466	0.127	0.740
Doctor	Nurse	0.797	3.093	0.091	0.637
Microsoft	Windows	Undef	3.559	0.541	0.783
Apple	Ipod	Undef	2.408	0.644	0.876

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

are. Resnik [21] defines the information shared by two terms as the maximum information content of the common parents of the terms in the ontology (Equation (10)).

$$Resnik(t_i, t_j) = \max_{t \in CP(t_i, t_j)} [-\log(p(t))] \quad (10)$$

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

Because the value of Equation (10) can vary between 0 to infinity, we use Lin’s metric instead [10], which varies between 0 (dissimilarity) and 1 (similarity).

$$Lin(t_i, t_j) = \frac{2 \times \max_{t \in CP(t_i, t_j)} [-\log(p(t))]}{IC(t_i) + IC(t_j)} \quad (11)$$

Table 7 shows semantic similarity and WebSim of selected terms. Semantic similarity is obtained by using WordNet::Similarity [18]. 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 Work

In order to accommodate dynamically changing knowledge, we presented a Web search engine based similarity framework that is referred to as WebSim. WebSim

is composed of two similarity metrics, an MI-based one, and a feature-based one. The formal is computationally cheap while the latter can produce more reliable similarity values. In addition, we suggested diverse ways on how WebSim can be utilized for ontology modification. Finally, coupling with semantic similarity, we demonstrated how WebSim is able to identify unknown relations in WordNet.

We intend to extend this work into the following three directions. First, we plan to study more sophisticated feature weighting schemes. Based on the observation that the Web page with high rank is generally more informative than the ones with low rank, each Web page can be weighted by order when features are extracted. That is, the features in the first page should be weighted higher than the features in the 20-th page, and so on. Second, it is worthwhile to study how different Web search engines affect WebSim. Finally, we plan to investigate the applicability of WebSim to ontology matching.

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. Castano, A. Ferrara, and S. Montanelli. H-MATCH: an algorithm for dynamically matching ontologies in peer-based systems. In *Proceedings of the 1st VLDB International Workshop on Semantic Web and Databases*, 2003.
3. 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.
4. S. Chung, and D. McLeod. Dynamic pattern mining: an incremental data clustering approach. *Journal on Data Semantics*, 2:85-112, 2005.
5. 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.
6. E.J. Glover, D.M. Pennock, S. Lawrence, and R. Krovetz. Inferring hierarchical descriptions. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2002.
7. 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.
8. J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 1998.
9. 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.
10. D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

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. M. Reinberger, P. Spyns, W. Daelemans, and R. Meersman. Mining for lexons: applying unsupervised learning methods to create ontology bases. In *Proceedings of International Conference on Ontologies, Databases and Applications of SEMantics*, 2003.
16. 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.
17. N.F. Noy, M. Sintek, S. Decker, M. Crubézy, R.W. Ferguson, and M.A. Musen. Creating and acquiring Semantic Web contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60-71, 2001.
18. 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.
19. 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.
20. M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130-137, 1980.
21. 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.
22. G. Salton and M.J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1983.
23. 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.
24. P. Spyns, and M. Reinberger. Lexically evaluating ontology triples generated automatically from texts. In *Proceedings of the 2nd European Semantic Web Conference*, 2005.
25. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: collaborative ontology development for the Semantic Web. In *Proceedings of International Semantic Web Conference*, 2002.
26. M. Ehrig, and Y. Sure. Ontology mapping - an integrated approach. In *Proceedings of the 1st European Semantic Web Symposium*, 2004.
27. C. Ziegler, G. Lausen, and L. Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2004.
28. Google Web APIs. <http://www.google.com/apis/>.