A Web-based Novel Term Similarity Framework for Ontology Learning

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*]*@pollux.usc.edu*

Abstract. Given that pairwise similarity computations are essential in ontology learning and data mining, we propose a similarity framework that 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 our proposed methodology, we present two different measures for similarity computation, a mutual information based and a feature-based metric. Moreover, we show how the proposed metrics can be utilized for modifying existing ontologies. Finally, we compare the extracted similarity relations with semantic similarity using WordNet. Experimental results show that our method can extract topical relations between terms that are not present 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 data becomes nearly impossible, the analysis would be performed by intelligent information management techniques to fulfill users' information requests quickly.

Representation and extraction of semantic meanings from information contents is essential in intelligent information management. This issue has been investigated 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 vague information needs or

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

⋆⋆ To whom correspondence should be addressed.

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 handcrafted ontologies such as WordNet [15] or CYC [11], 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 [19, 27], 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 efficiently build and maintain domain ontologies with the support of ontology learning. Given that computation of the similarity between terms is at the core of ontology learning, 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 framework), 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, for each term, we can obtain the freshest content 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 amounts of Web pages for feature extraction, since crawled one is a small subset of the entire Web contents, it still suffers from a data sparseness problem.

At the core of WebSim, we present two similarity metrics, an information-theoretic metric and a feature-based metric. The former one is a mutual information based measure that utilizes the number of Web pages associating with each term. In contrast, the latter one 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. We also show how ontologies can be modified with WebSim.

One of the main problems in concept-based ontologies is that topically related concepts and terms are not explicitly linked. For example, consider the Sports domain ontology that we developed in our previous work [9]. 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. Thus, conventional ontologies have a limitation in supporting a topical search. To address this problem, we also demonstrate how topical relations are generated using WebSim, and compare WebSim with semantic similarity in WordNet. In sum, the purpose of this research is to move one step forward to achieving the development of a novel similarity model that can be utilized for any ontology learning framework.

The remainder of this paper is organized as follows. In Section 2, we briefly review the related work, and highlight the strengths and weaknesses of 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. In Section 7, we present applications to which WebSim is applicable. Finally, we conclude the paper and provide our future plan in Section 8.

2 Related Work

The computation of the similarity between terms is at the core of ontology learning. There have been many attempts to determine similar term pairs from text corpora. One of the widely used approaches in similarity computation is based on distributional hypothesis [6, 21]. That is, if terms occur in a similar context, then they tend to have similar meanings. The context for a term t_i can be defined in diverse ways. For example, it can be represented by co-occurrence of words within grammatical relationships (e.g., verbs that take t_i as a subject or object, adjectives that modifies t_i, etc), or co-occurring words with t_i in a certain length of a window. Each context is referred to as features of a term.

Recently, there have been research efforts for building ontologies automatically [16, 14, 26, 1, 18, 7, 25]. In order to obtain context, they usually utilizes certain amount of crawled Web documents as corpus. Our approach is different from the previous work in that a Web search engine is employed to exploit the full content of the Web. Consequently, rather than relying on a small subset of the Web, we can access as much content as possible (depending on how many documents a search engine crawler can index). 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 because the extracted features or term similarity can be utilized for any ontology learning framework. Therefore, WebSim is expected to be a key enabling technique for the tasks where pairwise similarity computations play a central role.

Symbol	Definition
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
D_i	A set of top 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
v_i	A vector for i-th term
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 t_i
$count(t_i)$	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 used in this paper

3 WebSim: A Simple Mutual Information Approach

In this Section, we present a simple, yet powerful similarity metric. 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:

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

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

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 the 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

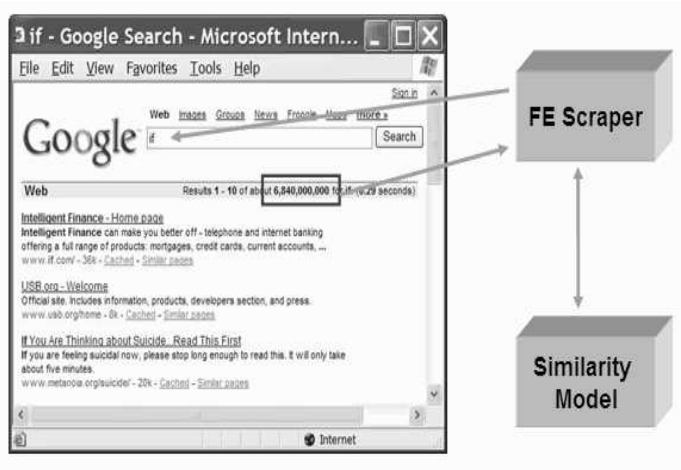

Fig. 1. Overview of an MI-based WebSim

query is matched with [3]. 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, $p(t_i)$ is defined as the number of documents returned by a search engine for a query t_i ($df(t_i)$) divided by total number of document indexed by a search engine (N).

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

Consequently, in contrast to previous text mining approaches, WebSim uses the different notion of $p(t_i)$.

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 in Table 2.

Table 2 presents sample results. As shown, in most cases (Type 1), WebSim captures similarity between related term pairs fairly well using both M_1 and M_5. 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*).

[3] 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.

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	*DAML*	*OIL*	2.89	7.28
	Notebook	*Computer*	0.05	4.38
	Operating System	Unix	1.54	5.28
	Text Mining	*Computational Biology*	3.70	-2.18
3	OWL	metadata	0.71	-3.23
	OWL	*OIL*	-0.16	-3.98
	Apple	*Computer*	-0.90	0.28
	Data Mining	Classification	1.75	1.27

Table 2. Mutual information for sample term pairs

This is expected in that M_1 provides more accurate context for a term than M_5 does. In some cases (Type 3), WebSim fails to capture similarity between terms. 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, there is a need to incorporate content (associated with the term) into WebSim, which motivates the necessity of a feature-based similarity metric.

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 based on the extracted features.

4.1 Feature Extraction

In this section, we explain how to extract features for each term. The notions of "term" and "word" are firsts defined. Although "term" and "word" generally have same meanings, for the convention, "term" will be used to refer to the entity of similarity computation (i.e., similarity is measured between terms) while "word" will be used to refer to a feature of "term". Extracting meaningful features for WebSim consists of the following three phases:

- Retrieval of Web documents for each term.
- Preprocessing of the retrieved Web documents.
- Construction of a vector space model using 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 [30] [4]. Consequently, for each t_i, a set of the top most relevant documents (D_i) is obtained by a search engine.

Next, meaningful information are extracted from Web pages in D_i using standard IR tools. This process includes HTML preprocessing (e.g., removing irrelevant HTML tags or Javascript code, etc), tokenization, stemming [22], stopwords removal, etc.

Finally, a term (t_i) is represented as a vector (v_i) in a vector space [24]. Toward this end, we employ a *bag-of-words* approach. That is, 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. In addition, TF can be normalized to reflect different document length. Let f_{ij} be the j-th feature of t_i, and $freq_{ijk}$

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

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:

$$tf_{ijk} = \frac{freq_{ijk}}{l_k} \tag{3}$$

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 WebSim, since only relevant documents with respect to a term are retrieved, if appropriate stopwords are removed in the preprocessing step, then 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:

$$w_{ij} = \frac{n_{ij}}{N_i} \times \frac{\Sigma_{d_k \in D_i} tf_{ijk}}{N_i} \tag{4}$$

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 top (high-weighted) few features contribute substantially to the norm of a vector, only high-weighted features (that make up most of the norm) are retained. This approach reduces the significant number of features while this minimizes the loss of information.

Table 3 shows sample features. Note that all features are stemmed (e.g., "inform" refers to "information"). As shown, 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, the next step is to measure closeness between vectors. Toward this end, we employ a Cosine metric that measures similarity of two vectors according to the angle between them [24]. The cosine of the angle between two vectors t_i and t_j is defined by

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

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 classif 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

Table 3. Sample features for terms

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 (dissimilar) to 1 (similar).

The underlying assumption of the proposed approach is simple yet effective: if t_i (e.g., data mining) and t_j (e.g., knowledge discovery) are similar, then the Web pages returned by t_i and t_j would be somewhat similar, consequently, Cosine value between v_i and v_j becomes high. Table 4 illustrates this. However, Sim_1 sometimes fails to capture similarity relations in case a term is ambiguous.

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 not coherent to a single sense of a term. 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 Web search. Because user queries usually tend to be short, the queries can be ambiguous, which often leads to irrelevant search results.

Table 5 shows top high-weighted features for three 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. Additionally, with regard to the top 50 pages for a query"oil" in Google, only 2 pages are about Ontology Inference Layer, and remaining pages are pointers to information on gas oil. Consequently, all extracted features for "oil" are related with gasoline. This is because Web search engines tend to rank a page based on popularity of a page using the notion of authorities and hubs [10, 2] as well as how a query is matched with the page (i.e., whether the query is matched with title, etc).

The extracted features for "oil" are useful ones if domain experts intend to add "oil" (in terms of energy context) into ontologies. However, if they consider OIL (Ontology Inference Layer) in Semantic Web context, then the extracted features are problematic. Assuming "oil" in an energy context is already in ontologies (because it was coined a long time ago), OIL in a Semantic Web context is of particular interest in terms of enriching ontologies (because it is neology).

In previous information retrieval research, query expansion has been widely studied in order to provide more useful search results. That is, a query can be refined by adding additional relevant search terms. The key point here 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

Table 5. Features for ambiguous terms

that the added terms should be somewhat related with the original query term. Otherwise, query expansion leads to a degradation of precision [9].

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 k 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 similarity (i.e., similarity between t_i and t_jt_i is expected to be high) [5]. Thus, if the similarity between t_i and t_j is not high enough, then the similarity will be refined as follows:

$$Sim_2(t_i, t_j) = Average(Cosine(v_i, v_{ji}),\ Cosine(v_j, v_{ij})) \tag{6}$$

where v_{ij} is a vector representation for the documents that are retrieved by issuing a query, t_it_j.

Table 6 shows how term expansion successfully refines 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 exists 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.

5 Ontology Modification with WebSim

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

[5] In this step, both t_it_j and t_jt_i are submitted as queries to the Web search engine because the order of query terms affects the search results.

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 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
Linux automata	linux *automata* program softwar version *cellular* simul org game *life* file comput window java *state model* 3d

Table 6. Sample features for terms with context

One of the key issues in ontology enrichment is how to identify candidate terms that should be added into an ontology. In our previous work, 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 [5, 4]. 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 a 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 be effectively utilized for 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 can be 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 [6]. If the obtained features do not exist in the ontology, then domain experts can add the features for the purpose of ontology enrichment. Alternatively, terms identified by topic mining can be given as input to WebSim to populate

[6] Since Porter stemmer is used in WebSim, it is difficult to perform reverse mapping from a stemmed word to original one. To address this problem, rather than relying on Porter stemmer, we can simply remove trailing "s" if the word is not in exception list (i.e., exception list contains words such as news).

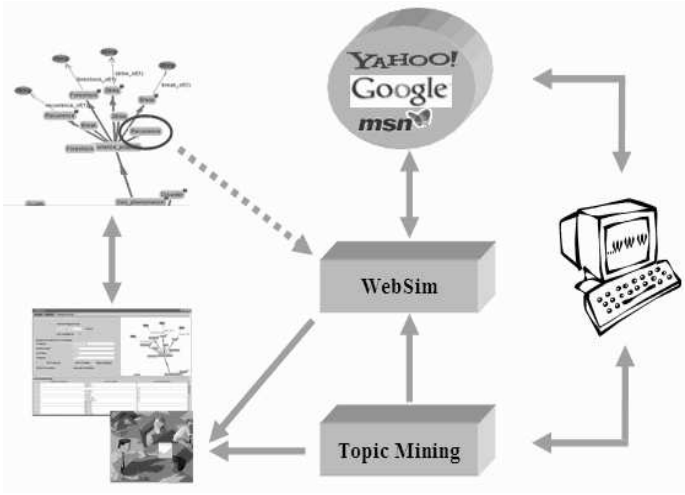

Fig. 2. Overview of the process for ontology modification with WebSim

features, which will be 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 changes. Toward this end, we present an WebSim-based approach to select candidate term pairs whose relationships 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. Because we cannot predict which part of the ontology should be modified (consider OIL and XML that are far from each other in conventional conceptual ontologies), we need $O(m^2)$ pairwise similarity computations if the size of the ontology is m. 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 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 average difference between the number of documents of $t_i t_j$ and that of $t_j t_i$ is 912.70. This number is negligible 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

$t_i t_j$). In contrast, for the feature-based WebSim, both $t_i t_j$ and $t_j t_i$ (as well as t_i and t_j) need to be submitted to a Web search engine. 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. However, the feature-based WebSim produces 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.

Given a pair of terms, t_i and t_j, a simple similarity measure in ontologies is to use an edge counting method. That is, the distance corresponds to the number of edges between terms in the ontology. 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. Consequently, it cannot provide correct similarity estimation.

Recently, semantic similarity metrics have been proposed to evaluate similarity between two terms in a taxonomy based on information content [12, 23]. These approaches rely on the incorporation of empirical probability estimates into a taxonomic structure. Previous studies have shown that these types of approaches are significantly less sensitive to link density variability. The notions of concept frequency and concept probability are first defined as follows:

$$concept\_freq(t_i) = \sum_{t_j \in C_{t_i}} count(t_j) \tag{7}$$

where C_{t_i} is the set of terms subsumed by a term t_i. Each term that occurs in the corpus is counted as an occurrence of each concept containing it.

$$prob(t_i) = \frac{concept\_freq(t_i)}{C} \tag{8}$$

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

The information content of a term t_i ($IC(t_i)$) can be quantified based on Equation (8).

$$IC(t_i) = -log(prob(t_i)) \tag{9}$$

Equation (9) 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 similarity between terms in ontology. The more information two terms share, the more similar they are. Resnik [23] defines the information shared by two terms as

the maximum information content of the common parents of the terms in the ontology (Equation (10)).

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

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 [12], which varies between 0 (maximum dissimilarity) and 1 (maximum similarity).

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

Table 7 shows semantic similarity and WebSim of selected terms. Semantic similarity is computed by using WordNet::Similarity [20]. 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 pair that was detected as highly similar by semantic similarity (e.g., "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 either the MI-based approach or refinement while uncovers topical relations that do not exist in WordNet.

7 Potential Applications of WebSim

In this Section, we explore sample applications to which WebSim is applicable.

- *Ontology matching.* Ontology matching, which aims at identifying mappings between related entities of multiple ontologies, has been widely studied recently [13, 3, 28]. 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 presents similarity metrics that are orthogonal to existing ones.
- *Recommendation Systems.* 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 [29]. 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 extend existing taxonomy maintained by online shopping malls (e.g., amazon.com).

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
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.

– *Term subspace clustering.* Subspace clustering aims at identifying clusters in subspaces of the original feature space. We recently developed an efficient subspace clustering algorithm that is scalable to the number of dimensions [8]. By coupling with the feature extraction methodology of WebSim, we can perform subspace clustering on terms such that each term can be assigned to different subspace clusters depending on the nature of the term (e.g., "clustering" can be assigned to both "data mining" and "computer architecture" clusters). Given that WebSim produces relevant features for a term with multiple meanings, WebSim can play a key role in term subspace clustering.

8 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 equipped with two similarity metrics, an MI-based one and a feature-based one. The former 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 will extend this work into the following four directions. First, although we demonstrated the usefulness of our feature extraction methodology, not all features of a term represent distinctive characteristics for the term. To address this problem, we will study the methodology on employing different search engines (i.e., a different feature set can be generated by each search engine), and taking intersection on feature sets obtained by multiple search engines. Consequently, the resulting feature set is expected to be more robust than the one with a single search engine. Second, we plan to study a sophisticated feature weighting scheme. Based on the observation that the Web page with higher rank is generally more informative than the ones with lower 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. Third, given that the MI-based and the feature-based similarity metrics may produce different similarity values for a same term pair, we plan to develop a new similarity measure to combine both metrics. Finally, we plan to investigate the applicability of WebSim to diverse applications such as ontology matching or term subspace clustering.

9 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. We also would like to thank the anonymous reviewers for their valuable comments.

References

1. E. Agirre, O. Ansa, E. Hovy, and D. Martinez. Enriching very large ontologies using the WWW. In *Proceedings of the ECAI Workshop on Ontology Learning*, 2000.
2. 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.
3. 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.
4. 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.
5. S. Chung, and D. McLeod. Dynamic pattern mining: an incremental data clustering approach. *Journal on Data Semantics*, 2:85-112, 2005.
6. 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.
7. 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.
8. J. Jun, S. Chung, and D. McLeod. Subspace clustering of microarray data based on domain transformation. To appear in *Proceedings of VLDB Workshop on Data Mining on Bioinformatics*, 2006.
9. 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.
10. J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 1998.
11. 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.
12. D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

13. J. Madhavan, P.A. Bernstein, A. Doan, and A.Y. Halevy. Corpus-based schema matching. In *Proceedings of the 21st International Conference on Data Engineering*, 2005.
14. A. Maedche, and S. Staab. Ontology learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2), 2001.
15. G. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235-312, 1990.
16. M. Missikoff, P. Velardi, and P. Fabriani. Text mining techniques to automatically enrich a domain ontology. *Applied Intelligence*, 18(3):323-340, 2003.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130-137, 1980.
23. 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.
24. G. Salton and M.J. McGill. *Introduction to modern information retrieval.* McGraw-Hill, 1983.
25. 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.
26. P. Spyns, and M. Reinberger. Lexically evaluating ontology triples generated automatically from texts. In *Proceedings of the 2nd European Semantic Web Conference*, 2005.
27. 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.
28. M. Ehrig, and Y. Sure. Ontology mapping - an integrated approach. In *Proceedings of the 1st European Semantic Web Symposium*, 2004.
29. 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.
30. Google Web APIs. http://www.google.com/apis/.