IJETCR Open Access

ISSN: 2348 - 2117

International Journal of Engineering Technology and Computer Research (IJETCR)
Available Online at www.ijetcr.org
Volume 2 Issue 4 Page No. 57-66

Semantic similarity measure using web page count, NGD and snippet based method

Mrs. Anita Jadhav, Prof. M. B. Ansari

Shreeyash College of Engineering & Technology, Aurangabad, Maharashtra, India.

Anita.jadhav17@gmail.com

ABSTRACT

Measuring the semantic similarity between words is an important component in various tasks on the web such as relation extraction, community mining, document clustering, and automatic metadata extraction. Despite the usefulness of semantic similarity measures in these applications, accurately measuring semantic similarity between two words (or entities) remains a challenging task. We propose an empirical method to estimate semantic similarity using page counts and text snippets retrieved from a web search engine for two words. Specifically, we define various word co-occurrence measures using page counts and integrate those with lexical patterns extracted from text snippets. To identify the numerous semantic relations that exist between two given words, we propose a novel pattern extraction algorithm and a pattern clustering algorithm. The optimal combination of page counts-based co-occurrence measures and lexical pattern clusters is learned using support vector machines. The proposed method outperforms various baselines and previously proposed web-based semantic similarity measures on three benchmark data sets showing a high correlation with human ratings. Moreover, the proposed method significantly improves the accuracy in a community mining task.

Keywords: web text analysis, web mining and information extraction.

INTRODUCTION:

Accurately measuring the semantic similarity between words is an important problem in web mining, information retrieval, and natural language processing. Web mining applications such as, community extraction, relation detection, and entity disambiguation; require the ability to accurately measure the semantic similarity between concepts or entities. In information retrieval, one of the main problems is to retrieve a set of documents that is semantically related to a given user query. Efficient estimation of semantic similarity between words is critical for various natural language processing tasks such as word sense disambiguation (WSD), textual entailment, and automatic text summarization. Semantically related words of a particular word are listed in manually created general-purpose lexical ontologies such as WordNet.1 In WordNet, a synset contains a set of synonymous words for a particular sense of a word. However, semantic similarity between entities changes over time and across domains. For example, apple is frequently associated with computers on the web. However, this sense of apple is not listed in most general-purpose thesauri or dictionaries. A user who searches for apple on the web, might be interested in this sense of apple and not apple as a fruit. New words are constantly being created as well as new senses are assigned to existing words. Manually maintaining ontologies to capture these new words and senses is costly if not impossible. We propose an automatic method to estimate the semantic similarity between words or entities using web page count, NGD (Normalized Google Distance) and Snippet based methods, because of the vastly numerous documents and the high growth rate of the web, it is time consuming to analyze each document separately. Web search engines provide an efficient interface to this vast information. Page counts, NGD and snippets are useful information sources provided by most web search engines. Page count of a query is an estimate of the number of pages that contain the query words. In general, page count may not necessarily be equal to the word frequency because the queried word might appear many times on one page. Page count for the query P AND Q can be considered as a global measure of co-occurrence of words P and Q. For example, the page count of the query “apple” AND “computer” in Google is 288,000,000, whereas the same for “banana” AND “computer” is only 3,590,000. The more than 80 times more numerous page counts for “apple” AND “computer” indicate that apple is more semantically similar to computer than is banana. Despite its simplicity, using

page counts alone as a measure of co-occurrence of two words presents several drawbacks. First, page count analysis ignores the position of a word in a page. Therefore, even though two words appear in a page, they might not be actually related. Second, page count of a polysemous word (a word with multiple senses) might contain a combination of all its senses. For example, page counts for apple contain page counts for apple as a fruit and apple as a company. Moreover, given the scale and noise on the web, some words might co-occur on some pages without being actually related. For those reasons, page counts alone are unreliable when measuring semantic similarity.

Snippets, a brief window of text extracted by a search engine around the query term in a document, provide useful information regarding the local context of the query term. Semantic similarity measures defined over snippets, have been used in query expansion, personal name disambiguation, and community mining. Processing snippets is also efficient because it obviates the trouble of downloading web pages, which might be time consuming depending on the size of the pages. However, a widely acknowledged drawback of using snippets is that, because of the huge scale of the web and the large number of documents in the result set, only those snippets for the pranking results for a query can be processed efficiently. Ranking of search results, hence snippets is determined by a complex combination of various factors unique to the underlying search engine. Therefore, no guarantee exists that all the information we need to measure semantic similarity between a given pair of words is contained in the top-ranking snippets. We propose a method that considers both page counts and lexical syntactic patterns extracted from snippets that we show experimentally to overcome the above mentioned problems. The phrase is the largest indicates a hypernymic relationship between Jaguar and cat. Phrases such as also known as, is a, part of, is an example of all indicate various semantic relations. Such indicative phrases have been applied to numerous tasks with good results, such as hypernym extraction and fact extraction. From the previous example, we form the pattern X is the largest Y, where we replace the two words Jaguar and cat by two variables X and Y.

Our contributions are summarized as follows:

• We present an automatically extracted lexical syntactic patterns-based approach to compute the semantic similarity between words or entities using text snippets retrieved from a web search engine. We propose a lexical pattern extraction algorithm that considers word sub-sequences in text snippets. Moreover, the extracted sets of patterns are clustered to identify the different patterns that describe the same semantic relation.

• We integrate different web-based similarity measures using a machine learning approach along with NGD method. We extract synonymous word pairs from Word Net synsets as positive training instances and automatically generate negative training instances. We then train a two-class support vector machine (SVM) to classify synonymous and non-synonymous word pairs. The integrated measure outperforms all existing web based semantic similarity measures on a benchmark data set.

• We apply the proposed semantic similarity measure to identify relations between entities, in particular people, in a community extraction task. In this experiment, the proposed method outperforms the baselines with statistically significant precision and recall values. The results of the community mining task show the ability of the proposed method to measure the semantic similarity between not only words, but also between named entities, for which manually created lexical ontologies do not exist or incomplete.

RELATED WORK

Given taxonomy of words, a straightforward method to calculate similarity between two words is to find the length of the shortest path connecting the two words in the taxonomy. If a word is polysemous, then multiple paths might exist between the two words. In such cases, only the shortest path between any two senses of the words is considered for calculating similarity. A problem that is frequently acknowledged with this approach is that it relies on the notion that all links in the taxonomy represent a uniform distance. Resnik proposed a similarity measure using information content.

Resnik defined the similarity between two concepts C1 and C2 in the taxonomy as the maximum of the information content of all concepts C that subsume both C1 and C2. Then, the similarity between two words is defined as the maximum of the similarity between any concepts that the words belong to. He used WordNet as the taxonomy; information content is calculated using the Brown corpus. Semantic similarity measures have been used in various applications in natural language processing such as word sense disambiguation, language modeling, synonym extraction, and automatic thesauri extraction. Semantic similarity measures are important in many web related tasks. In query expansion, a user query is modified using synonymous words to improve the relevancy of the search. One method to find appropriate words to include in a query is to compare the previous user queries using semantic similarity measures. If there

exists a previous query that is semantically related to the current query, then it can be either suggested to the user, or internally used by the search engine to modify the original query.

OUTLINE METHOD:

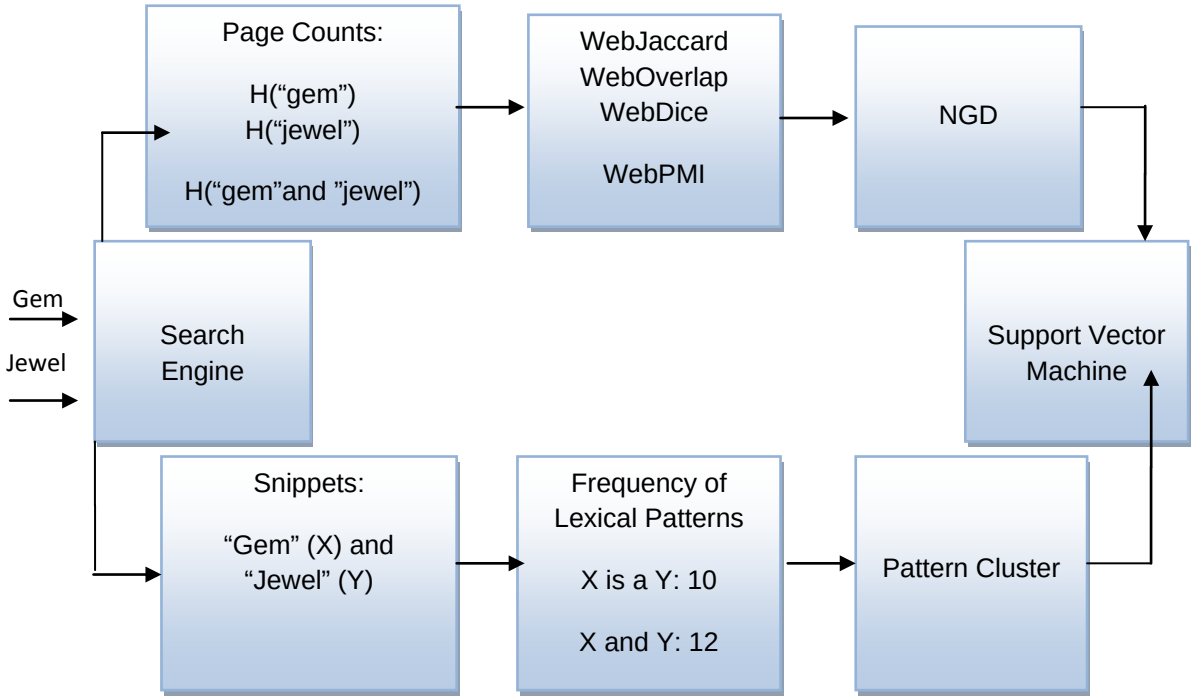

Figure 1: Outline Diagram of Proposed Method

3.1 Modules in Proposed System:

- Page Count-Based Co-Occurrence Measures and NGD Method
- Lexical Pattern Extraction
- Lexical Pattern Clustering
- Measuring Semantic Similarity
- SVM Training
- Miller-Charles Benchmark Data Set

3.3.1 Page Count-Based Co-Occurrence Measures:

The Page count method [4] is widely used to measure the semantic similarity between two words, the actual page counts for the query words gets retrieved from the web search engine. The page counts for the query P and Q can be considered as an approximation of co-occurrence of two words (or multiword phrases) P and Q on the web. However, page counts for the query P and Q alone do not accurately express semantic similarity. For example, Google returns 11,300,000 as the page count for "car" and "automobile," whereas the same is 49,000,000 for "car" and "apple." Although, automobile is more semantically similar to car than apple is, page counts for the query "car" and "apple" are more than four times greater than those for the query "car" and "automobile." One must consider the page counts not just for the query P and Q, but also for the individual words P and Q to assess semantic similarity between P and Q.

The four popular co-occurrence measures are used for page count WebJaccard, WebOverlap (Simpson), WebDice, and Web PMI (Pointwise mutual information), to compute semantic similarity using page counts. Here use the notation $H(P)$ to denote the page counts for the query P in a search engine and $H(Q)$ to denote the page counts for the query Q in a search engine.

WebJaccard Method:

The WebJaccard coefficient between words P and Q, is defined as

$$WebJaccard(P,Q) = \begin{cases} 0, & if\ H(P \cap Q) \le C, \\ \dfrac{H(P \cap Q)}{H(P) + H(Q) - H(P \cap Q)}, & otherwise. \end{cases} \quad \text{..... (3.1)}$$

In equation 3.1, $H(P \cap Q)$ denotes the conjunction query P and Q. Given the scale and noise in web data, it is possible that two words may appear on some pages even though they are not related. In order to reduce the adverse effects attributable to such co-occurrences, set the WebJaccard coefficient to zero if the page count for the query $H(P \cap Q)$ is less than a threshold C.

WebOverlap Method:

The WebOverlap coefficient between words P and Q, is defined as

$$WebOverlap(P,Q) = \begin{cases} 0, & if\ H(P \cap Q) \le C, \\ \dfrac{H(P \cap Q)}{Min(H(P), H(Q))}, & otherwise. \end{cases} \quad \text{..... (3.2)}$$

WebDice Method:

The WebDice coefficient between words P and Q, is defined as

$$WebDice(P,Q) = \begin{cases} 0, & if\ H(P \cap Q) \le C, \\ \dfrac{2H(P \cap Q)}{H(P) + H(Q)}, & otherwise. \end{cases} \quad \text{..... (3.3)}$$

WebPMI Method:

The WebPMI coefficient between words P and Q, is defined as

$$WebPMI(P,Q) = \begin{cases} 0, & if\ H(P \cap Q) \le C, \\ \log_2 \left(\dfrac{\frac{H(P \cap Q)}{N}}{\frac{H(P)}{N} \frac{H(Q)}{N}} \right), & otherwise. \end{cases} \quad \text{..... (3.4)}$$

Point wise mutual information is a measure that is motivated by information theory; it is intended to reflect the dependence between two probabilistic events. Here, in equation 3.4, N is the number of documents indexed by the search engine. Probabilities are estimated according to the maximum likelihood principle. To calculate PMI accurately, N should be known, the number of documents indexed by the search engine. Although estimating the number of documents indexed by a search engine is an interesting task itself, it is beyond the scope of this work. In the present work, set N as ¼ 10^{10} according to the number of indexed pages reported by Google. As previously discussed, page counts are mere approximations to actual word co-occurrences in the web. However, it has been shown empirically that there exists a high correlation between word counts obtained from a web search engine (e.g., Google and AltaVista) and that from a corpus (e.g., British National corpus).

NGD (Normalized Google Distance) Method:

Cilibrasi and Vitanyi [14] proposed a distance metric between words using only page counts retrieved from a web search engine. This proposed metric is named Normalized Google Distance (NGD) and is given by below formula

$$NGD(P,Q)=\frac{G(x,y)-\min(G(x),G(y))}{\max(G(x),G(y))}$$
$$=\frac{\max\{\log f(x),\log f(y)\}-\log f(x,y)}{\log N-\min\{\log f(x),\log f(y)\}} \quad \text{..... (2.8)}$$

In equation 2.8, $f(x)$ denotes the number of pages containing x, and $f(x,y)$ denotes the number of pages containing both x and y, as reported by Google. This NGD is an approximation to the NID of using the prefix code-word lengths (Google code) generated by the Google distribution as defining a compressor approximating the length of the Kolmogorov code, using the background knowledge on the web as viewed by Google as conditional information.

3.3.2 Lexical Pattern Extraction:

In this module, distances between two query words are extracted from the snippets. The page count based method can be problematic if one or both words are polysemous, or when page counts are unreliable. On the other hand, the snippets returned by a search engine for the conjunctive query of two words provide useful clues related to the semantic relations that exist between two words. A snippet [8] contains a window of text selected from a document that includes the queried words. Snippets are useful for search because, most of the time, a user can read the snippet and decide whether a particular search result is relevant, without even opening the URL. Using snippets as contexts is also computationally efficient because it obviates the need to download the source documents from the web, which can be time consuming if a document is large. For example, consider the below snippet.

"Cricket *is a* ***sport*** *played between two teams, each with eleven players."*

Here, the phrase *is a* indicates semantic relationship between cricket and sport. Many such phrases indicate semantic relationships. For example, also known as, is a, part of, is an example of all indicate semantic relations of different types. In the example given above, words indicating the semantic relation between cricket and sport appear between the query words. Replacing the query words by variables X and Y, form the pattern X is a Y from the example given above.

Despite the efficiency of using snippets, they pose two main challenges: first, a snippet can be a fragmented sentence, second, a search engine might produce a snippet by selecting multiple text fragments from different portions in a document. Because most syntactic or dependency parsers assume complete sentences as the input, deep parsing of snippets produces incorrect results. Consequently, here lexical pattern extraction algorithm has been proposed using web snippets, to recognize the semantic relations that exist between two words. Lexical syntactic patterns have been used in various natural language processing tasks such as extracting hypernyms or meronyms question answering and paraphrase extraction. Although a search engine might produce a snippet by selecting multiple text fragments from different portions in a document, a predefined delimiter is used to separate the different fragments. For example, in Google, the delimiter "..." is used to separate different fragments in a snippet.

Given two words P and Q, query a web search engine using the wildcard query "P _ _ _ _ _ Q" and download snippets. The "_" operator matches one word or none in a webpage. Therefore, wildcard query retrieves snippets in which P and Q appear within a window of seven words. Because a search engine snippet contains 20 words on average, and includes two fragments of texts selected from a document, assume that the seven word window is sufficient to cover most relations between two words in snippets. In fact, over 95 percent of the lexical patterns extracted by the proposed method contain less than five words.

Attempt to approximate the local context of two words using wildcard queries.

For a snippet S, retrieved for a word pair (P,Q), first, replace the two words P and Q, respectively, and with two variables X and Y. Next, generate all subsequence's of words from δ that satisfy all of the following conditions:

1. A subsequence must contain exactly one occurrence of each X and Y.
2. The maximum length of a subsequence is L words.
3. A subsequence is allowed to skip one or more words. However, do not skip more than g number of words consecutively. Moreover, the total number of words skipped in a subsequence should not exceed G.
4. Expand all negation contractions in a context. For example, didn't is expanded to did not. Do not skip the word not when generating subsequence's. For example,

this condition ensures that from the snippet X is not Y, do not produce the subsequence X is a Y.

Here a new Lexical pattern extraction algorithm is proposed which will extract all the patterns from the snippets.

Lexical Pattern Extraction Algorithm:

Input: Snippets S returned by the web Search engine for query words $H(P)$ and $H(Q)$.

Output: Distance (D) between query word $H(P)$ and $H(Q)$

Step 1: Read each snippet S

Step 2: for each snippet S do

if $H(P)$ word found in snippets then set index as 0 value end if

if $H(Q)$ word found in snippets then set index as (0++) value count the distance between $H(P)$ and $H(Q)$ i.e. D end if

if ASCII char for Enter then go-to Step 3

else go-to Step 2 end for

Step 3: for each snippet S do

Record the distance between $H(P)$ and $H(Q)$ in database end for

Step 4: Return $H(P)$, $H(Q)$ & Distance (D)

Finally, count the frequency of all generated subsequences and only use subsequences that occur more than 1times as lexical patterns. It is noteworthy that the proposed pattern extraction algorithm considers all the words in a snippet, and is not limited to extracting patterns only from the mid fix (i.e., the portion of text in a snippet that appears between the queried words). Moreover, the consideration of gaps enables us to capture relations between distant words in a snippet. Specifically, use the constraints (1-7) to prune the search space of candidate subsequences. For example, if a subsequence has reached the maximum length 7 then it will not extend it further.

3.3.3 Lexical Pattern Clustering:

Typically, a semantic relation can be expressed using more than one pattern. For example, consider the two distinct patterns, X is a Y, and X is a large Y. Both these patterns indicate that there exists and is-a relation between X and Y. Identifying the different patterns that express the same semantic relation enables us to represent the relation between two words accurately. According to the distributional hypothesis, words that occur in the same context have similar meanings. The distributional hypothesis has been used in various related tasks, such as identifying related words, and extracting paraphrases. If it consider the word pairs that satisfy (i.e., co-occur with) a particular lexical pattern as the context of that lexical pair, then from the distributional hypothesis, it follows that the lexical patterns which are similarly distributed over word pairs must be semantically similar.

Represent a pattern a by a vector a of word-pair frequencies. Designate, the word-pair frequency vector of pattern a. It is analogous to the document frequency vector of a word, as used in information retrieval. The value of the element corresponding to a word pair (Pi,Qi) in a, is the frequency $F(Pi,Qi,a)$, that the pattern a occurs with the word pair (Pi,Qi). As demonstrated later, the proposed pattern extraction algorithm typically extracts a large number of lexical patterns. Clustering algorithms based on pairwise comparisons among all patterns are prohibitively time consuming when the patterns are numerous. Next, a sequential clustering algorithm is discussed to efficiently cluster the extracted patterns.

Algorithm: Sequential pattern clustering algorithm.

Input: patterns Λ= {a1... an}, threshold θ

Output: Clusters C

1: SORT (Λ)
2: C←{}
3: for pattern ai∈Λ do 4: max←-∞
5: c*←null
6: for cluster cj∈C do 7: sim← cosine (ai, cj)
8: if sim>max then
9: max←sim 10: c*←cj
11: end if
12: end for
13: if max>θ then
14: c*←c*⊕ai
15: else
16: C←C∪{ai}
17: end if
18: end for
19: return C

Given a set Λ of patterns and a clustering similarity threshold θ, Algorithm 1 returns clusters (of patterns) that express similar semantic relations. First, in Algorithm 1, the function SORT sorts the patterns into descending order of their total occurrences in all word pairs. The total occurrence μ(a) of a pattern a is the sum of frequencies over all word pairs

$$\mu(a)=\sum_{i} F(Pi,Qi,a) \quad \text{..... (3.5)}$$

After sorting, the most common patterns appear at the beginning in Λ, whereas rare patterns (i.e., patterns that

occur with only few word pairs) get shifted to the end. Next, in line 2, initialize the set of clusters, C to the empty set. The outer for loop (starting at line 3), repeatedly takes a pattern ai from the ordered set Λ, and in the inner for loop (starting at line 6), finds the cluster, $c*(\in C)$ that is most similar to ai. First, represent a cluster by the centroid of all word-pair frequency vectors corresponding to the patterns in that cluster to compute the similarity between a pattern and a cluster. Next, compute the cosine similarity between the cluster centroid (cj), and the word-pair frequency vector of the pattern (ai). If the similarity between a pattern ai, and its most similar cluster, $c*$ is greater than the threshold θ, append ai to $c*$. The operator $\oplus$ denotes the vector addition between $c*$ and ai. Then, form a new cluster $\{ai\}$ and append it to the set of clusters C, if $\{ai\}$ is not similar to any of the existing clusters beyond the threshold θ.

By sorting the lexical patterns in the descending order of their frequency and clustering the most frequent patterns first, form clusters for more common relations first. This enables us to separate rare patterns which are likely to be outliers from attaching to otherwise clean clusters. The greedy sequential nature of the algorithm avoids pairwise comparisons between all lexical patterns. This is particularly important because when the number of lexical patterns is large as in this experiments (e.g., over 100,000), pairwise comparisons between all patterns are computationally prohibitive. The proposed clustering algorithm attempts to identify the lexical patterns that are similar to each other more than a given threshold value. By adjusting the threshold, obtain clusters with different granularity.

The similarity threshold θ ranges in 0 to 1. It decides the purity of the formed clusters. Setting θ to a high value ensures that the patterns in each cluster are highly similar. However, high θ values also yield numerous clusters (increased model complexity). Experimentally, the effect of θ on the overall performance of the proposed relational similarity measure.

The sequential nature of the algorithm avoids pairwise comparisons among all patterns. Moreover, sorting the patterns by their total word-pair frequency prior to clustering ensures that the final set of clusters contains the most common relations in the data set.

3.3.4 Measuring Semantic Similarity:

We have seen above page count co-occurrence measures for page counts and NGD, further Lexical pattern extraction and lexical pattern clustering will be studied to extract clusters of lexical patterns from snippets to represent numerous semantic relations that exist between two words. In this module, a machine learning approach is discussed to combine both page counts-based co-occurrence measures, and snippets-based lexical pattern clusters to construct a robust semantic similarity measure.

Given N clusters of lexical patterns, first, represent a pair of words (P,Q) by an $(N+1)$ dimensional feature vector fpq. The four page counts-based co-occurrence measures defined in Section 3.2.2 are used as four distinct features in fpq. For completeness, let us assume that $(N+1)st, (N+2)nd, (N+3)rd, and\ (N+4)th$ features are set, respectively, to WebJaccard, WebOverlap, WebDice, and WebPMI. Next, compute a feature from each of the N clusters as follows: first, assign a weight Wij to a pattern ai that is in a cluster cj as follows:

$$wij = \frac{\mu(ai)}{\sum_{t \in cj}^{n} \mu(t)} \quad \text{..... (3.7)}$$

In equation 3.7, μ_a is the total frequency of a pattern an in all word pairs. Because to perform a hard clustering on patterns, a pattern can belong to only one cluster (i.e. $W_{ij} = 0\ for\ ai \notin cj$). Finally, compute the value of the j th feature in the feature vector for a word pair (P,Q) as follows:

$$\sum_{ai \in cj} Wij\ f(P,Q,ai) \quad \text{..... (3.8)}$$

The value of the j th feature of the feature vector Fpq representing a word pair (P,Q) can be seen as the weighted sum of all patterns in cluster Cj that co-occur with words P and Q. assume all patterns in a cluster to represent a particular semantic relation. Consequently, the jth feature value above equation expresses the significance of the semantic relation represented by cluster j for word pair (P,Q). For example, if the weight Wij is set to 1 for all patterns ai in a cluster Cj, then the j th feature value is simply the sum of frequencies of all patterns in cluster Cj with words P and Q. However, assigning an equal weight to all patterns in a cluster is not desirable in practice because some patterns can contain misspellings and/or can be grammatically incorrect. Assigns a weight to a pattern proportionate to its frequency in a cluster. If a pattern has a high frequency in a cluster, then it is likely to be a canonical form of the

relation represented by all the patterns in that cluster. Consequently, the weighting scheme described by equation prefers high frequent patterns in a cluster.

Train a two-class SVM to detect synonymous and nonsynonymous word pairs, utilize a training data set $S=\{(Pk,Qk,Yk)\}$ of word pairs. S consists of synonymous word pairs (positive training instances) and nonsynonymous word pairs (negative training instances). Training data set S is generated automatically from Word Net synsets. Label $Y_k \in\{-1,1\}$ indicates whether the word pair (P_k,Q_k) is a synonymous word pair (i.e. $Y_k=1$) or a nonsynonymous word pair (i.e. $Y_k=-1$). For each word pair in S, create a $(N+4)$ dimensional feature vector as described above. To simplify the notation, let us denote the feature vector of a word pair (P_k,Q_k) by F_k. Finally, train a two-class SVM using the labeled feature vectors.

An SVM will be trained using synonymous and nonsynonymous word pairs and will be used to compute the semantic similarity between two given words. Following the same method will be used to generate feature vectors for training, create a $(N+4)$ dimensional feature vector $F*$ for a pair of words $(P*,Q*)$, to measure semantic similarity.

3.5 SVM Training:

To train the two-class SVM described above, this required both synonymous and nonsynonymous word pairs, WordNet have been used, WordNet is a manually created English dictionary, to generate the training data required by the proposed method. For each sense of a word, a set of synonymous words is listed in WordNet synsets. Randomly selected 3,000 nouns from WordNet, and extracted a pair of synonymous words from a synset of each selected noun. If a selected noun is polysemous, then considered the synset for the dominant sense. Obtaining a set of nonsynonymous word pairs (negative training instances) is difficult, because there does not exist a large collection of manually created nonsynonymous word pairs.

Consequently, to create a set of nonsynonymous word pairs, adopt a random shuffling technique. Specifically, first randomly select two synonymous word pairs from the set of synonymous word pairs created above, and exchange two words between word pairs to create two new word pairs. For example, from two synonymous word pairs (A,B) and (C,D), generate two new pairs (A,C) and (B,D). If the newly created word pairs do not appear in any of the word net synsets, select them as nonsynonymous word pairs. Repeat this process until create 3,000 nonsynonymous word pairs, final training data set contains 6,000 word pairs (i.e., 3,000 synonymous word pairs and 3,000 nonsynonymous word pairs). Next, I used the lexical pattern extraction algorithm to extract numerous lexical patterns for the word pairs in training data set. Because of the noise in web snippets such as, ill-formed snippets and misspells, most patterns occur only a few times in the list of extracted patterns. Consequently, ignore any patterns that occur less than 1 times. Finally, de-duplicate the patterns that appear for both synonymous and nonsynonymous word pairs to create a final set of lexical patterns.

3.6. Miller-Charles Benchmark Data Set:

The proposed semantic similarity measure using web page count, NGD and snippet has been evaluated by comparing it with human ratings of Miller-Charles (MC: 28 pairs) benchmark dataset. A semantic similarity measure is evaluated using the correlation between the similarity scores produced by it for the word pairs in a benchmark data set and the human ratings. Both Pearson correlation coefficient and Spearman correlation coefficient have been used as evaluation measures in previous work on semantic similarity. It is noteworthy that Pearson correlation coefficient can get severely affected by nonlinearities in ratings. Contrastingly, Spearman correlation coefficient first assigns ranks to each list of scores, and then computes correlation between the two lists of ranks. Therefore, Spearman correlation is more appropriate for evaluating semantic similarity measures, which might not be necessarily linear. Most semantic similarity measures are nonlinear. However, for the MC data set, which contains only 28 word pairs, Pearson correlation coefficient has been widely used. To be able to compare results with previous work, use both Pearson and Spearman correlation coefficients for experiments conducted on MC data set.

RESULT OF THE PROPOSED SYSTEM:

Query 1	Query 2	Web Jaccard	Web Overlap	Web Dice	Web PMI	Similarity
Cord	Smile	0.007	0.068	0.032	0.325	0.318
Rooster	Voyage	0.119	0.104	0.206	0.43	0.33
Noon	String	0.06	0.017	0.118	0.414	0.469
Glass	magician	0.03	0.05	0.07	0.316	0.329
Monk	Slave	0.085	0.057	0.157	0.565	0.338
Coast	Forest	0.381	0.108	0.513	0.837	0.704
Monk	Oracle	0.03	0.029	0.07	0.037	0.321
Lad	Wizard	0.014	0	0.044	0.206	0.327
Forest	graveyard	0.006	0.189	0.03	0.215	0.326
Food	Rooster	0	0.002	0.019	0.574	0.336
Coast	Hill	1	0.35	0.942	1	1
Car	Journey	0.482	0.185	0.605	0.598	0.311
Crane	Implement	0.192	0.086	0.303	0.21	0.329
Brother	Lad	0.156	0.076	0.257	0.136	0.329
Bird	Crane	0.086	0.26	0.158	0.296	0
Bird	Cock	0.024	0.014	0.061	0.316	0.33
Food	Fruit	0.336	0.339	0.469	0.592	0.33
Brother	Monk	0.119	0.212	0.206	0.393	0.33
Asylum	Madhouse	0.026	0.058	0.064	0	0.323
Furnace	Stove	0.007	0.01	0.031	0.383	0.345
Magician	Wizard	0.511	0.321	0.63	0.622	0.415
Journey	Voyage	0.144	0.138	0.241	0.797	1
Coast	Shore	0.345	0.196	0.478	0.528	0.33
Implement	Tool	0.174	0.501	0.28	0.459	0.7
Boy	Lad	0.039	0.142	0.084	0.386	0.329
Automobile	Car	0.877	1	0.877	0.578	1
Midday	Noon	0.036	0.067	0.081	0.143	0.329
Gem	Jewel	0.145	0.16	0.243	0.174	0.299

CONCLUSIONS:

We proposed a semantic similarity measure using page counts, NGD and snippets retrieved from a web search engine for two words, four word co-occurrence measures were computed using page counts. We proposed a lexical pattern extraction algorithm to extract numerous semantic relations that exist between two words. Moreover, a sequential pattern clustering algorithm was proposed to identify different lexical patterns that describe the same semantic relation. Both page counts-based co-occurrence measures and lexical pattern clusters were used to define features for a word pair. A two-class SVM was trained using those features extracted for synonymous and non-synonymous word pairs selected from WorldNet synsets. Experimental results on three benchmark data sets showed that the proposed method outperforms various baselines as well as previously proposed web-based semantic similarity measures, achieving a high correlation with human ratings. Moreover, the proposed method improved the F-score in a community mining task, thereby underlining its usefulness in real-world tasks that include named entities not adequately covered by manually created resources.

REFERENCES:

1. M. Li, X. Chen, X. Li, B. Ma, and P. Vitanyi, "The Similarity Metric," *IEEE Trans. Information Theory*, vol. 50, no. 12,Dec. (2004), pp. 3250-3264.

2. D. Mclean, Y. Li, and Z.A. Bandar, "An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources," *IEEE Trans. Knowledge and Data Eng*., vol. 15, no. 4 , July/Aug. (2003), pp. 871-882.

3. A. Gledson and J. Keane, "Using Web-Search Results to Measure Word-Group Similarity," *Proc. Int'l Conf. Computational Linguistics (COLING '08)*, (2008), pp. 281-288.

4. D. Bollegala, Y. Matsuo &M. Ishizuka, "A web search engine-based approach to measure semantic similarity between words", *IEEE Transactions on Knowledge and Data Eng*., 2011, Vol. 23, No.7, (2011), pp977-990.

5. G. Miller and W. Charles, "Contextual Correlates of Semantic Similarity," *Language and Cognitive Processes*, vol. 6, no. 1, (1998) ,pp. 1-28.

6. G. Hirst and D. St-Onge, "Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms," WordNet: An Electronic Lexical Database, MIT Press, (1998) , pp. 305-332.

7. G. A. Miller, "WordNet: A lexical database for english", Comm. ACM, Vol. 38, No. 11, (1995), pp39-41. http://www.cogsci.princeton.edu/wn

8. D. Bollegala, Y. Matsuo, and M. Ishizuka, (2007) "Measuring semantic similarity between words using web search engines", Proceeding of International Conference on World Wide Web, (2007), pp757-766.

9. D. Bollegala, Y. Matsuo &M. Ishizuka,"Disambiguating personal names on the web using automatically extracted key phrases", Proceeding of 17th European Conference on Artificial Intelligence, (2006), pp. 553-557.

10. R. Rada, H. Mili, E. Bichnell, and M. Blettner, "Development and Application of a Metric on Semantic Nets", IEEE Trans. Systems, Man and Cybernetics, vol. 19, no. 1, Jan./Feb. (1989), pp.17-30.

11. P. Resnik, "Semantic Similarity in a Taxonomy: An Information Based Measure and Its Application to Problems of Ambiguity in Natural Language," J. Artificial Intelligence Research, vol. 11, (1999), pp. 95-130.

12. D. Lin, "Automatic Retrieval and Clustering of Similar Words," Proc. 17th Int'l Conf. Computational Linguistics (COLING), (1998), pp. 768- 774.

13. J. Jiang & D. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy", Proceeding of International Conference on Research in Computational Linguistics (ROCLING X), (1997), pp.117- 124.

14. R. Cilibrasi and P. Vitanyi, "The Google Similarity Distance," IEEE Trans. Knowledge and Data Eng., vol. 19, no. 3, Mar. (2007), pp. 370-383.

15. M. Sahami & T. Heilman,"A web-based kernel function for measuring the similarity of short text snippets", Proceeding of 15th International World Wide Web Conference, (2006), pp. 1-18.

16. H. Chen, M. Lin, and Y. Wei, "Novel Association Measures Using Web Search with Double Checking," Proc. 21st Int'l Conf. Computational Linguistics and 44th Ann. Meeting of the Assoc. for Computational Linguistics (COLING/ACL '06), (2006), pp. 1009-1016.

[17] F. Keller and M. Lapata, "Using the Web to Obtain Frequencies for Unseen Bigrams," Computational Linguistics, vol. 29, no. 3, (2003), pp. 459- 484.

17. R. K. Srihari, Z. F. Zhang & A. B. Rao, "Intelligent indexing and semantic retrieval of multimodal documents", Information Retrieval, Vol. 2, (2000), pp. 245-275.

18. J. Pei, J. Han, B. Mortazavi-Asi, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Mining Sequential Patterns by Pattern- Growth: The Prefixspan Approach," IEEE Trans. Knowledge and Data Eng., vol. 16, no. 11, Nov. (2004), pp. 1424-1440.

19. A. Bagga and B. Baldwin, "Entity-Based Cross Document Coreferencing Using the Vector Space Model," Proc. 36th Ann. Meeting of the Assoc. for Computational Linguistics and 17th Int'l Conf. Computational Linguistics (COLING-ACL), (1998), pp. 79-85.