International Journal of Engineering Research and Development
***e**-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com*
Volume 3, Issue 5 (August 2012), PP. 62-71

A Novel Approach To Measure Semantic Similarity Between Words Using Web Search Engine

Chandrakala.K[1], Prof. E.V.Prasad[2]
[1]*Department of Computer Science & Engineering, UCEK, JNTUKAKINADA, Kakinada, Andhra Pradesh, INDIA*
[2]*Professor of Computer Science & Engineering & Registrar-JNTUK, Kakinada, Andhra Pradesh, INDIA*

Abstract—Semantic similarity measures play important roles in Information Retrieval (IR) and Natural Language Processing (NLP). Accurately measuring the semantic similarity between two words (or entities) is an important problem in web mining. Web mining application such as community mining, relation detection, and entity disambiguation requires the ability to accurately measure the semantic similarity between concepts or entities remains a challenging task. We propose a novel approach to estimate semantic similarity that uses the information available on the Web to measure similarity between words or entities. The proposed method exploits page counts and text snippets returned by a Web search engine. We define various similarity scores for two given words W_1 and W_2, using the page counts for the queries W_1, W_2 and W_1 AND W_2. Moreover, we propose a novel approach to compute semantic similarity using automatically extracted lexico-syntactic patterns from text snippets. These different similarity scores are integrated using support vector machines, to leverage a robust semantic similarity measure. The proposed method significantly improves the accuracy comparatively the existing methods.

Keywords- Web mining, information extraction, Semantic Similarity, web text analysis.

I. INTRODUCTION

The study of semantic similarity between words has long been an integral part of information retrieval and natural language processing. Semantic similarity between entities changes over time and across domains. For example, *blackberry* is frequently associated with *phones* on the Web. However, this sense of *blackberry* is not listed in most general-purpose thesauri or dictionaries. A user, who searches for *blackberry* on the Web, may be interested in this sense of *blackberry* and not *blackberry* as a fruit. New words are constantly being created as well as new senses are assigned to existing words. Manually maintaining thesauri to capture these new words and senses is costly if not impossible. We propose an automatic method to measure semantic similarity between words or entities using Web search engines. Because of the vastly numerous documents and the high growth rate of the Web, it is difficult to analyze each document separately. Web search engines provide an efficient interface to this vast information. Page counts and Snippets are two useful information sources provided by most Web search engines. Page count of a query is the number of pages that contain the query words. Page count for the query W_1 *AND* W_2 can be considered as a global measure of co-occurrence of words W_1 and W_2. For example, the page count of the query *"blackberry"* AND *"phone"* in *Google* is 510, 000, 000, whereas the same for *"strawberry"* AND *"phone"* is only 84, 200, 000. The more than 70 times more numerous page counts for *"blackberry"* AND *"phone"* indicate that *blackberry* is more semantically similar to *phone* than is *strawberry*. Despite its simplicity, using page counts alone as a measure of co-occurrence of two words presents several draw backs. First, page count analyses ignore the position of a word in a page. Therefore, even though two words appear in a page, they might not be actually related. Secondly, page count of a polysemous word (a word with multiple senses) might contain a combination of all its senses. For an example, page counts for *apple* contain page counts for *apple* as a fruit and *apple* as a company. Moreover, given the scale and noise in the Web, some words might occur arbitrarily, i.e. by random chance, on some pages. 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 [2], personal name disambiguation [3] and community mining [4]. Processing snippets is also efficient as 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 top-ranking 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.

This paper proposes a method that considers both page counts and *lexico-syntactic patterns* extracted from snippets, thereby overcoming the problems described above. For example, let us consider the following snippet from Google for the query *Lion AND cat*.

*"The **Lion** is the largest **cat** found in Western Africa and usually hunts in coordinated groups and stalks their chosen prey"*

Fig.1. A snippet retrieved for the query ***Lion*** *AND* ***cat***

Here, the phrase is the largest indicates a hypernymic relationship between Lion 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 [5] and fact extraction [6]. From the previous example, we form the pattern X is the largest Y, where we replace the two words Lion 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 subsequences 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. We extract synonymous word pairs from WordNet 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 nonsynonymous word pairs. The integrated measure outperforms all existing web based semantic similarity measures on a benchmark data set.

II. 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 [7]. 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 [8] proposed a similarity measure using information content. He defined the similarity between two concepts C_1 and C_2 in the taxonomy as the maximum of the information content of all concepts C that subsume both C_1 and C_2. 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. Li et al. [9] combined structural semantic information from a lexical taxonomy and information content from a corpus in a nonlinear model. They proposed a similarity measure that uses shortest path length, depth, and local density in taxonomy. Their experiments reported a high Pearson correlation coefficient of 0.8914 on the Miller and Charles [10] benchmark data set. They did not evaluate their method in terms of similarities among named entities. Lin [11] defined the similarity between two concepts as the information that is in common to both concepts and the information contained in each individual concept.

Sahami and Heilman [2] measured semantic similarity between two queries using snippets returned for those queries by a search engine. For each query, they collect snippets from a search engine and represent each snippet as a TF-IDF-weighted term vector. Each vector is L2 normalized and the centroid of the set of vectors is computed. Semantic similarity between two queries is then defined as the inner product between the corresponding centroid vectors. They did not compare their similarity measure with taxonomy-based similarity measures.

Chen et al. [4] proposed a double-checking model using text snippets returned by a web search engine to compute semantic similarity between words. For two words W_1 and W_2, they collect snippets for each word from a web search engine. Then, they count the occurrences of word W_1 in the snippets for word W_2 and the occurrences of word W_2 in the snippets for word W_1. These values are combined nonlinearly to compute the similarity between W_1 and W_2. The Co-occurrence Double-Checking (CODC) measure is defined as

$$CODC\,(W_1, W_2) = \begin{cases} 0, & f(W_1@W_2) = 0 \\ \exp\left(log\left[\dfrac{f(W_1@W_2)}{H(W_1)} \times \dfrac{f(W_2@W_1)}{H(W_2)}\right]^{\alpha}\right), & otherwise \end{cases}$$

Here, $f(W_1@W_2)$ denotes the number of occurrences of W_1 in the top-ranking snippets for the query W_2 in Google, H (W_1) is the page count for query W_1, and α is a constant in this model, which is experimentally set to the value 0.15. This method depends heavily on the search engine's ranking algorithm. Although two words W_1 and W_2 might be very similar, we cannot assume that one can find W_2 in the snippets for W_1, or vice versa, because a search engine considers many other factors besides semantic similarity, such as publication date (novelty) and link structure (authority) when ranking the result set for a query. This observation is confirmed by the experimental results in their paper which reports zero similarity scores for many pairs of words in the Miller and Charles [10] benchmark data set. Semantic similarity measures have been used in various applications in natural language processing such as word sense disambiguation [14], language modeling [15], synonym extraction [16], and automatic thesauri extraction [17]. Semantic similarity measures are important in many web related tasks. In query expansion [18], 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.

D.Bollegala.et.al [34], measured semantic similarity between two words using snippets and page counts returned by a search engine. He defined four different Page co-occurrence metrics to calculate semantic similarity and combined these features with the features extracted from Snippets.

III. METHOD

A. Problem Definition and Outline

Given two words W_1 and W_2, we model the problem of measuring the semantic similarity between W_1 and W_2, as a one of constructing a function Sem*Sim* (W_1, W_2) that returns a value in range [0, 1]. If W_1 and W_2 are highly similar (e.g., synonyms), we expect Sem*Sim* (W_1, W_2) to be closer to 1. On the other hand, if W_1 and W_2 are not semantically similar, then we expect Sem*Sim* (W_1, W_2) to be closer to 0. We define numerous features that express the similarity between W_1 and W_2 using page counts and snippets retrieved from a web search engine for the two words. Using this feature representation of words, we train a two-class support vector machine to classify synonymous and nonsynonymous word pairs. The function Sem*Sim* (W_1, W_2) is then approximated by the confidence score of the trained SVM.

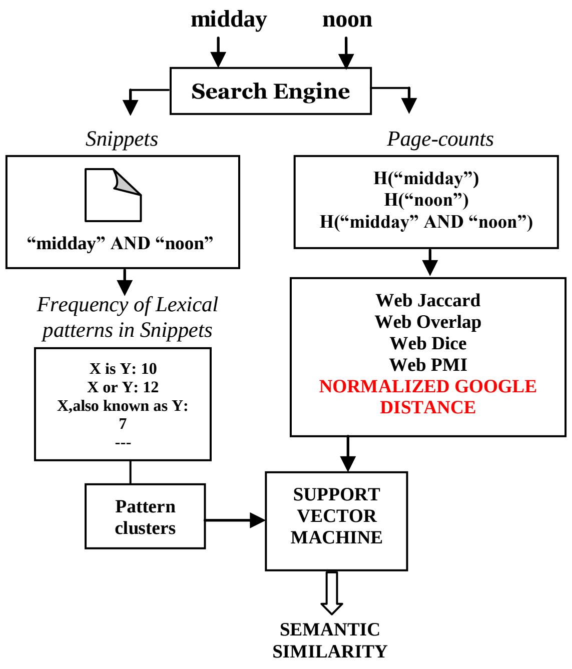

Fig. 2. *Outline of the proposed method*

Fig. 2 illustrates an example of using the proposed method to compute the semantic similarity between two words, midday and noon. First, we query a web search engine and retrieve page counts for the two words and for their conjunctive (i.e., "midday" "noon," and "midday AND noon"). In Section *B*, we define five similarity scores using page counts. Page counts-based similarity scores consider the global co-occurrences of two words on the web. However, they do not consider the local context in which two words co-occur. On the other hand, snippets returned by a search engine represent the local context in which two words co occurs on the web. Consequently, we find the frequency of numerous lexical syntactic patterns in snippets returned for the conjunctive query of the two words. The lexical patterns we utilize are extracted automatically using the method described in Section *C*. However, it is noteworthy that a semantic relation can be expressed using more than one lexical pattern. Grouping the different lexical patterns that convey the same semantic relation, enables us to represent a semantic relation between two words accurately. For this purpose, we propose a sequential pattern clustering algorithm in Section *D*. Both page counts-based similarity scores and lexical pattern clusters are used to define various features that represent the relation between two words. Using this feature representation of word pairs, we train a two-class support vector machine [19] in Section *E*.

B. Page Count-Based Co-Occurrence Measures

Page counts for the query W_1 AND W_2 can be considered as an approximation of co-occurrence of two words (or multiword phrases) W_1 and W_2 on the web. However, page counts for the query W_1 AND W_2 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 W_1 AND W_2, but also for the individual words W_1 and W_2 to assess semantic similarity between W_1 and W_2. We compute five popular co-occurrence measures; Jaccard, Overlap (Simpson), Dice, and Pointwise mutual information (PMI), Normalized Google Distance, to compute semantic similarity using page counts. For the remainder of this paper, we use the notation H (W_1) to denote the page counts for the query W_1 in a search engine. The WebJaccard coefficient between words (or multiword phrases) W_1 and W_2, WebJaccard (W_1, W_2) is defined as

$$Webjaccard(W_1,W_2) = \begin{cases} 0, & H(W_1 \cap W_2) \le c, \\ \dfrac{H(W_1 \cap W_2)}{H(W_1) + H(W_2) - H(W_1 \cap W_2)}, & otherwise \end{cases} \quad ----(1)$$

Therein, $W_1 \cap W_2$ denotes the conjunction query W_1 AND W_2. 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, we set the WebJaccard coefficient to zero if the page count for the query $W_1 \cap W_2$ is less than a threshold c.2 Similarly, we define Web Overlap, (W_1, W_2) as

$$Web\ Overlap(W_1,W_2) = \begin{cases} 0, & H(W_1 \cap W_2) \le c, \\ \dfrac{H(W_1 \cap W_2)}{min[H(W_1), H(W_2)]}, & otherwise \end{cases} \quad ----(2)$$

Web Overlap is a natural modification to the Overlap (Simpson) coefficient. We define the Web Dice coefficient as a variant of the Dice coefficient. Web Dice (W_1, W_2) is defined as

$$Web\ Dice(W_1,W_2) = \begin{cases} 0, & H(W_1 \cap W_2) \le c, \\ \dfrac{H(W_1 \cap W_2)}{H(W_1) + H(W_2)}, & otherwise \end{cases} \quad ----(3)$$

Point wise mutual information [20] is a measure that is motivated by information theory; it is intended to reflect the dependence between two probabilistic events. We define WebPMI as a variant form of point wise mutual information using page counts as

$$Web\ PMI(W_1,W_2) = \begin{cases} 0, & H(W_1 \cap W_2) \le c, \\ log\left(\dfrac{\frac{H(W_1 \cap W_2)}{N}}{\frac{H(W_1)}{N}\frac{H(W_2)}{N}}\right), & otherwise \end{cases} \quad ----(4)$$

Here, N is the number of documents indexed by the search engine. Probabilities in (4) are estimated according to the maximum likelihood principle. To calculate PMI accurately using (4), we must know N, the number of documents indexed by the search engine. Although estimating the number of documents indexed by a search engine [21] is an interesting task itself, it is beyond the scope of this work. In the present work, we set N= 10^{10} according to the number of indexed pages reported by Google. Normalized Google Distance (NGD) is a distance metric between words and is defined as

$$NGD(W_1,W_2) = \frac{max\{\log H(W_1), logH(W_2)\} - \log H(W_1,W_2)}{\log N - min\{\log H(W_1), logH(W_2)\}} \quad --- (5)$$

NGD is based on normalized information distance [13] which is defined using kolmogorov complexity. 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) [22]. Moreover, the approximated page counts have been successfully used to improve a variety of language modeling tasks [23].

C. Extracting Lexio- Syntactic Patterns from Snippets

Page counts-based co-occurrence measures described in Section *B* do not consider the local context in which those words co-occur. This 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 contains a window of text selected from a document that includes the queried words. For example, consider the snippet in Fig. 3. Here, the phrase is a indicates a 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.

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

Fig.3. Snippet retrieved for the query "Cricket and sport"

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, we can 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, we propose a shallow lexical pattern extraction algorithm 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 [5], [24], or meronyms [25], question answering [26], and paraphrase extraction [27]. 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. We use such delimiters to split a snippet before we run the proposed lexical pattern extraction algorithm on each fragment. Given two words P and Q, we 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, our wildcard query retrieves snippets in which P and Q appear within a window of seven words. Because a search engine snippet contains approximately 20 words on average, and includes two fragments of texts selected from a document, we 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. We attempt to approximate the local context of two words using wildcard queries. For example, Fig. 4 shows a snippet retrieved for the query "Toyoto**** Nissan"

*"**Toyoto** and **Nissan** are two major Japanese car manufacturers"*

Figure.4. Snippet retrieved for the query "Toyoto**** Nissan"

For a snippet δ, retrieved for a word pair *(W_1, W_2)*, first, we replace the two words W_1 and W_2, respectively, with two variables X and Y. We replace all numeric values by D, a marker for digits. Punctuation marks are not removed. Next, we generate all subsequences of words from δ by using modified Prefix Span algorithm.

Algorithm 1 *(Prefix Span) Prefix-projected sequential pattern mining.*
Input: A sequence database S, and the minimum support threshold min_support.
Output: The complete set of sequential patterns
Method: Call *Prefix Span ((), 0, S)*
Subroutine *Prefix Span (α, l, S/ α)*
The parameters are 1) α is a sequential pattern; 2) l is
the length of ; and 3) S/ α is the α -projected database if $\alpha \neq \langle\rangle$, otherwise, it is the sequence database S.
Method:

```
1. Scan S/ $\alpha$ once, find each frequent item, b, such that
    (a) b can be assembled to the last element of $\alpha$ to form a sequential pattern; or
    (b) ‹b› can be appended to $\alpha$ to form a sequential pattern.
2. For each frequent item b, append it to form a sequential pattern $\alpha^1$, and output $\alpha^1$.
3. For each $\alpha^1$, construct $\alpha$ $^1$-projected database S/ $\alpha^1$ 0, and call Prefix Span($\alpha^1$, l+1, S/ $\alpha^1$).
```

Finally we consider the subsequences 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, we 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. We expand all negation contractions in a context. For example, didn't is expanded to did not. We do not skip the word not when generating subsequences. For example, this condition ensures that from the snippet X is not a Y, we do not produce the subsequence X is a Y.

Lastly, we count the frequency of all generated subsequences and only use subsequences that occur more than T times as lexical patterns. We experimentally set the parameters L, g, G, and T to 5, 2, 4 and 5 respectively. 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). For example, some of the patterns extracted from the snippets shown in Fig 4 are: X, a large Y, X a flightless Y, and X, large Y lives.

D. Pattern Clustering Algorithms

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 an 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. We represent a pattern p by a vector **a** of word-pair frequencies. We designate **a**, the word-pair frequency vector of pattern p. 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 (W_{1_i}, W_{2_i}) in **a**, is the frequency, f(W_{1_i}, W_{2_i}, a), that the pattern **a** occurs with the word pair (W_{1_i}, W_{2_i}). Next, we present a greedy sequential clustering algorithm to efficiently cluster the extracted patterns.
Given a set **P** of patterns and a clustering similarity threshold θ, Algorithm 2 returns clusters (of patterns) that express similar semantic relations. First, in Algorithm 2, the function SORT sorts the patterns into descending order of their total occurrences in all word pairs. The total occurrence μ(p) of a pattern p is the sum of frequencies over all word pairs, and is given by

$$\mu(p) = \sum_i f(W_{1_i}, W_{2_i}, p) \quad \text{-----(6)}$$

After sorting, the most common patterns appear at the beginning in **P**, whereas rare patterns (i.e., patterns that occur with only few word pairs) get shifted to the end. Next, in line 2, we initialize the set of clusters, C, to the empty set. The outer for loop (starting at line 3), repeatedly takes a pattern $\mathbf{p_i}$ from the ordered set P, and in the inner for loop (starting at line 6), finds the cluster, c^* (ϵ C) that is most similar to p_i. First, we 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, we compute the cosine similarity between the cluster centroid (c_j), and the word-pair frequency vector of the pattern (p_i). If the similarity between a pattern p_i, and its most similar cluster, c^*, is greater than the threshold θ, we append p_i to c^* (line 14). We use the operator $\oplus$ to denote the vector addition
between c^* and p_i. Then, we form a new cluster $\{p_i\}$ and append it to the set of clusters, C, if p_i is not similar to any of the existing clusters beyond the threshold θ.

Algorithm 2. Greedy Sequential pattern clustering algorithm.

```
Input: patterns Ᵽ = { $p_1, p_{2, \ldots\ldots}, p_n$ } , threshold θ
Output: clusters C
    1: SORT(Ᵽ)
    2: C ← {}
    3: for pattern $p_i$ ϵ Ᵽ do
    4: max ←-∞
    5: c*←null
    6: for cluster $c_j$ ϵ C do
    7: sim← cosine($p_i$,$c_j$)
    8: if sim > max then
    9: max ← sim
    10: c* ← $c_j$
    11: end if
    12: end for
    13: if max > θ then
    14: c* ← c* $\oplus$ $p_i$
    15: else
    16: C ← C $\bigcup$ {$p_i$}
    17: end if
    18: end for
    19: return C
```

By sorting the lexical patterns in the descending order of their frequency and clustering the most frequent patterns first, we 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. The only parameter in Algorithm 2, the similarity threshold θ, ranges in [0, 1]. It decides the purity of the formed clusters. So, we experimentally set theta value to 0.7 which is optimal. 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.

E. Measuring Semantic Similarity using support vector machines

Support Vector Machines are currently among the best performers for a number of classification tasks ranging from text to genomic data. SVMs can be applied to complex data types beyond feature vectors (eg. Graphs, sequences and relational data) by designing kernel functions for such data. SVM was trained using features of page count based co-occurrence measures and lexical pattern clustering to predict the synonymous & non-synonymous word pairs.

In section B, we defined five co-occurrence measures using page counts. Moreover, in sections C and D, we showed how to extract clusters of lexical patterns from snippets to represent numerous semantic relations that exist between two words. In this section we describe a machine learning approach to combine both page counts-based co-occurrence measures to construct a robust semantic similarity measure.

Given N clusters of lexical patterns, first, we represent a pair of words *(W_1, W_2)* by an (N+5)-dimensional feature vector$f_{W_1W_2}$. The five page count-based co-occurrence measures defined in section B are used for five different features in $f_{W_1W_2}$. For comprehensiveness, let us assume that (N+1)st, (N+2)nd, (N+3)rd, (N+4)th and (N+5)th features are set, respectively, to WebJaccard, WebOverlap, WebDice, WebPMI and Normalized Google Distance. Next we compute a feature from each of the N clusters as follows: first we assign a weight ω_{ij} to a pattern p_i that is in a cluster c_j as follows

$$\omega_{ij} = \frac{\mu(p_i)}{\sum_{t\epsilon c_j}\mu(t)} \qquad ----(7)$$

Here $\mu(p)$ is the total frequency of a pattern p in all word pairs, and it is given by (6). Because we perform a hard clustering on patterns, a pattern can belong to only one cluster (i.e. $\omega_{ij} = 0, for\, p_i \notin c_j$). Finally we compute the value of the jth feature vector for a word pair *(W_1, W_2)* as follows

$$\sum_{p_i \in c_j} \omega_{ij} f(W_{1_i}, W_{2_i}, p_i) \qquad ----- (8)$$

The value of the jth feature of the feature vector $f_{W_1W_2}$ representing a word pair *(W_1, W_2)* can be seen as the weighted sum of all patterns in cluster c_j that co-occur with words *W_1* and *W_2*. We assume all patterns in a cluster to represent a particular semantic relation. Consequently, the jth feature value given by (8) expresses the significance of the semantic relation represented by cluster j for word pair *(W_1, W_2)*.

To train a two-class SVM to detect synonymous and nonsynonymous word pairs, we utilize a training data set $S = \{(W_{1_k}, W_{2_k}, y_k)\}$ of word pairs. S consists of synonymous Word pairs (positive training instances) and non synonymous word pairs (negative training instances). Training data set S is generated automatically from Word Net synsets. We randomly select 2,500 nouns from Wordnet and extract synonymous words and non synonymous word pairs for each selected noun. Our final training data set contains 5,000 word pairs. Label $y_k \in \{-1, 1\}$ indicates whether the word pair *(W_{1k},W_{2k})* is a synonymous word pair (i.e.,y_k = 1)or a non synonymous word pair (i.e., ,y_k = -1). For each word pair in S, we create an (N+5)-dimensional feature vector as described above. To simplify the notation, let us denote the feature vector of a word pair *(W_{1k}, W_{2k})* by f_k. Finally, we train a two-class SVM using the labeled feature vectors. Once we have trained an SVM using synonymous and non synonymous word pairs, we can use it to compute the semantic similarity between any two given words. Following the same method, we used to generate feature vectors for training, we create an (N+5)-dimensional feature vector f^* for a pair of words *(W_1^*,W_2^*)*, between which we must measure semantic similarity. We define the semantic similarity SemSim(W_1^*, W_2^*) between W_1^* *and* W_2^* as the posterior probability, $p(y^* = 1|\boldsymbol{f}^*)$that the feature vector $\boldsymbol{f}^*$ corresponding to the word pair *(W_1^*, W_2^*)* belongs to the synonymous-words class (i.e., $\boldsymbol{y}^*$= 1). SemSim(W_1^*, W_2^*) is given by

$$SemSim(W_1^*, W_2^*) = p(y^* = 1|\boldsymbol{f}^*) \quad ----(\mathbf{9})$$

Because SVMs are large margin classifiers, the output of an SVM is the distance from the classification hyperplane. The distance d($\boldsymbol{f}^*$) to an instance $\boldsymbol{f}^*$ from the classification hyperplane is given by

$$d(\boldsymbol{f}^*) = \boldsymbol{h}(\boldsymbol{f}^*) + \boldsymbol{b}$$

Here,b is the bias term and the hyperplane, $\boldsymbol{h}(\boldsymbol{f}^*)$ is given by

$$\boldsymbol{h}(\boldsymbol{f}^*) = \sum_{\boldsymbol{i}} \boldsymbol{y_k \alpha_k K}(\boldsymbol{f_k}, \boldsymbol{f}^*) \text{ is}$$

Here, $\boldsymbol{\alpha_k}$ is the Lagrange multiplier corresponding to the support vector f_k, and $\boldsymbol{K}(\boldsymbol{f_k}, \boldsymbol{f}^*)$ is the value of the kernel function for a training instance f_k and the instance to classify, f^*. However, d(f^*) is not a calibrated posterior probability. Following Platt [30], we use sigmoid functions to convert this un calibrated distance into a calibrated posterior probability.

The probability, $p(y = 1\backslash d(f))$, is computed using a sigmoid

Table-1 : SEMANTIC SIMILARITY SCORES ON DATA SET

Word Pair	WebJaccard	WebDice	WebOverlap	WebPMI	NGD	D.Bollegala[34]	Proposed
Journey – voyage	0.79	0.88	1.00	1.00	0.10	1.00	1.00
gem – jewel	0.05	0.09	0.10	1.00	0.60	0.82	1.00
bird – cock	0.03	0.07	0.07	0.02	0.99	0.87	0.99
monk-oracle	0.00	0.01	0.03	0.30	0.95	0.80	0.95
boy – lad	0.02	0.04	0.43	1.00	0.82	0.96	1.00
automobile – car	0.07	0.14	0.78	0.66	0.84	0.92	0.84
car - journey	0.07	0.13	0.56	0.19	0.94	0.17	0.94
coast – shore	0.17	0.29	0.53	1.00	0.48	0.97	1.00
food - rooster	0.00	0.00	0.25	0.00	1.00	0.02	0.00
asylum – madhouse	0.00	0.01	0.09	1.00	0.66	0.79	1.00
magician-wizard	0.02	0.05	0.14	1.00	0.66	1.00	1.00
crane – implement	0.02	0.04	0.06	1.00	0.77	0.06	0.02
midday – noon	0.04	0.08	1.00	1.00	0.31	0.99	1.00

furnace – stove	0.04	0.08	0.11	1.00	0.56	0.88	0.93
food – fruit	0.14	0.24	0.84	1.00	0.68	0.94	0.95
implement – tool	0.08	0.15	0.58	1.00	0.65	0.50	1.00
bird – crane	0.03	0.06	0.17	1.00	0.79	0.85	1.00
brother – monk	0.03	0.06	0.30	1.00	0.74	0.27	1.00
brother – lad	0.03	0.06	0.31	1.00	0.73	0.13	0.02
coast – hill	0.20	0.33	0.43	1.00	0.57	0.36	0.20
forest – graveyard	0.01	0.02	0.23	1.00	0.81	0.44	0.01
monk – slave	0.02	0.04	0.08	1.00	0.74	0.24	0.02
coast- forest	0.19	0.31	0.34	1.00	0.50	0.15	0.19
lad – wizard	0.00	0.01	0.02	0.07	0.98	0.23	0.00
cord – smile	0.13	0.23	0.78	1.00	0.52	0.01	0.13
noon – string	0.02	0.04	0.08	0.42	0.92	0.00	0.02
rooster-voyage	0.00	0.00	0.03	0.00	0.43	0.05	0.00

function defined over d(f) as follows:

$$p(y = 1\backslash d(f)) = \frac{1}{1 + \exp(\lambda d(f) + \mu)}$$

Here, λ and μ are parameters which are determined by maximizing the likelihood of the training data. Log likelihood of the training data is given by

$$L(\lambda, \mu) = \sum_{k=1}^{N} \log p(y_k \backslash f_k; \lambda, \mu)$$

$$= \sum_{k=1}^{N} \{t_k \log(p_k) + (1 - t_k) \log(1 - p_k)\} ----(10)$$

Here, to simplify the notation, we have used $t_k = (y_k+1)/2$ and $p_k = p(y_k = 1\backslash f_k)$. The maximization in (10) with respect to parameters λ and μ is performed using model-trust minimization [31].

IV. CONCLUSION

In this paper, we proposed a measure that uses both page counts and snippets to robustly calculate semantic similarity between two given words. The method consists of five page-count-based similarity scores and automatically extracted lexico-syntactic patterns. We integrated different page-count based similarity scores with lexico syntactic pattern clusters using support vector machines. Training data were automatically generated using WordNet synsets. Proposed method outperformed all the baselines including previously proposed Web-based semantic similarity measures on a benchmark dataset. Only 2500 positive examples and 2500 negative examples are necessary to leverage the proposed method, which is efficient and scalable because it only processes the snippets (no downloading of Web pages is necessary) for the top ranking results by Google. A contrasting feature of our method compared to the WordNet based semantic similarity measures is that our method requires no taxonomies, such as Word Net, for calculation of similarity. The results are given in Table-1. Table 1 shows the experimental results on Miller Charles data set for the proposed method (Proposed) and previously proposed web based semantic similarity measures.All similarity scores in Table-1are normalized into [0,1] range for the ease of comparison.Therefore, the proposed method can be applied in many tasks where such taxonomies do not exist or are not up-to-date. Results of our experiments indicate that the proposed method can robustly capture semantic similarity between words.

REFERENCES

[1]. A.Kilgarriff, “Googleology is Bad Science, “Computational Linguistics, vol. 33,pp. 147-151,2007.

[2]. M.sahami and T.Heilman, “a Web- Based Kernel Function for Measuring the Similarity of short Text Snippets, “proc.15 th Int’l World Wide Web Comf.,2006.

[3]. D.Bollegala, Y.Matsuo, and M.Ishizuka, “Disambiguating Personal Names on the Web Using automatically Extracted Key Phrases,” Proc.17 th European Comf. Artificial Intelligence, PP.553-557,2006.

[4]. H.Chern,M.Lin, and Y.Wei, “Novel Association Measures Using Web Search with Double Checking, “Proc.21 th Int’l Conf.Computational Linguistics and 44 th ann. Meeting of the Assoc, for Computational linguistics (COLING/ACI..’06),PP. 1009-1016,2006.

[5]. M.hearst, “Automatic Acquisition of Hyponyms from Large Text Corpora, “Proc,14 th Conf. Computational Linguistics (COLING),pp.539-545,1992.

[6]. M.Pasca, D.Lin,J.Bigham, A.Lifchits, and A. Jain, “Organizing and Searching the World Wide Web of Facts-Step One: The OneMillion Fact Extraction Challenge ,”Proc. Nat” l Comf. Artificial Intelligence (AAAI ’06), 2006.

[7]. R.Rada, H.Mili, E. Bichnel, and M.Blettner,” Development and application of a Metric on Semantic Nets, ”IEEE Trans Systems, Man and Cybernetics vol.19,no.1,pp.17-30, Jan./Feb.1989.

[8]. P.Resnik, “Using Information Content to Evaluate Semantic similarity in a Taxonomy,” Proc.14 th Int’l Joint Conf.Aritificial intelligence, 1995.

[9]. D.Mclean, Y.Li, and Z.A Bandar, “An Approach for Measuring Semantic similarity between Words using Multiple Information Sources, “IEEE Trans.Knowledge and Date Eng,.vol.15 ,no 4,pp.871-882, July /Aug.2003.

[10]. G.Miller and W.Charles, “Contextual Correlates of Semantic similarity,”Language and Cognitive Processes, vol.6,no,l,pp.1-28,1998.

[11]. D.Li,”An Information. Theoretic Definition of Similarity,”Proc.15 th Int’l Conf.Machine Learning (ICML),PP.296-304,1998.

[12]. R.Cilibrasi and P.Vitanyi,”The Google Similarity Distance, “IEEETrans.Knowledge and data Eng., vol.19, no.3, pp.370-383,Mar.2007.

[13]. M.Li,X.Chen, X.Li, B.Ma, and P.Vitanyi, “The similarity Metri,”IEEE Trans.Information Theory,Vol.50,no.12,pp.3250-3264,Dec.2004.

[14]. P.Resnik,”Semantic Similarity in a taxomomy: An Information Based Measure and Its Application to Problems of ambiguity in Natural Language,”I.Artificial Intelligence Research, vol.11.PP.95-130,1999.

[15]. R.Rosentield, “A.Maximum Entropy approach to adaptive Statistical Modeling,”computer Speech and Language, vol.10,pp.187-228,1996.

[16]. D.Lin,”Automatic Retrieval and Clustering of similar Words,” Proc.17 th Int’l Conf.Coputational Linguistics (COLIG),pp.768-774,1998.

[17]. J.Curran,”Ensemble Methods for Automatic Thesaurus Exptration,”Proc.ACL-02 Conf.Empirical Methods in Natural Language Processing (EMNLP),2002.

[18]. C.Buckley, G.salton, J.Allan,and A.Singhal,”Automatic Query Expansion Using smart:Trec.3,”Proc.Third Text Retrieval Conf., PP.69-80,1994.

[19]. V.Vapnik, statistical Learning Theory, Wiley,1998.

[20]. K.Church and P.Hanks,”Work association Norms, Mutual Information and Lexicography,”Computational Linguistics, vol.16,pp .22-29,1991.

[21]. Z.Bar-Yossef and M.Gurevich, “Random sampling from a search Engine’s Index,” Proc. 15 th Int’l World Wide Web Conf.,2006.

[22]. F.Keller and M.Lapata,”Using the Web to Obtain Frequencies for Uneen Bigrams, “Computational Linguistics, vol.29,no.3,pp.459-484.2003.

[23]. M.Latata and f.keller, ”Web-Based Models for Natural Language Pdrocessing,” ACM Trans. Speech and Language Processing. Vol 2,no 1,pp.1-31,2005.

[24]. R.Snow,D Jurafsky, and A.Ng,”Learning syntactic Patterns for Automatic Hypermym Discovery, “proc.Advances in Neural Information Processing systems (NIPS),pp.1297-1304,2005.

[25]. M.Berland and E.Charniak. “Finding parts in Very Large Corpora,”Proc. Ann. Meeting of the Assoc. for Computational Linguitics on Computational Linguistics (ACL’99),pp.57-64,1999.

[26]. D.Ravichandran and E.Hovy,”learning Surface Text Patterns for a Question answering system,” Proc. Ann. Meeting on Assoc.for Computational Linguistics (ACL’02),pp.4-47,2001.

[27]. R.Bhagat and d.Ravichandran, “Large Scale Acquisition of Paraphrases for Learning Surface patterns,”Proc.Assoc .for Computational Linguistics .Human Language Technologies (ACL’08:HLT),pp.674-682,2008.

[28]. 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 perfixspan Approach,” IEEE Trans.Knowledge and data Eng.,vol .16,no.11,pp.1424-1440,Nov.2004.

[29]. Z.Harris,”Distributional structure,”Word,vol.10,pp.146-162,1954.

[30]. J.Platt,”probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods,” Advance in Large Margin Classifiers, pp.61-74.MIT Press,2000.

[31]. P.Gill, W.Murray, and M.Wright, Practical Optimization. Academic Press, 1981.

[32]. H.Rubenstein and J.Goodenough,”Contextual conrelates of synonymy,” Comm.ACM.vol.8,pp.627-633,1965.

[33]. L.Finkelstein, E.Gabrilovich,Y.Matias, E.Rivlin, Z.solan, G.Wolfman, and E.Ruppin, " Placing Search in Context:The concept Revisited,"ACM Tran.Information Systems, vol.20.pp.116-131,2002.

[34]. D.Bollegala, Y.Matsuo,and M.Ishizuka, "Measuring Semantic Similarity between Words Using Web Search Engines,"Proc. Inf'l Conf.World Wide Web (WWW'07), pp.757-766,2007.

[35]. M.Strube and S.P.Ponzetto, "Wikirelate! Computing Semantic Relatedness Using Wikipedis,"Proc.Nat'l Conf.Aritificial intelligence (AAAI'06),pp.1419-1424.2006.

[36]. A.Gledson and J.Keane,"Using web-Search Results to Measure Word-Group similarity,"Proc.Int'l Conf.Computational Linguistics (COLING '08),pp.281.2008.

[37]. Z. Wu and M. Palmer, "Verb Semantics and Lexical Selecton," Proc. Ann. Meeting on Assoc. for Computational Linguistics (ACL '94), PP. 133-138, 1994.

[38]. C. Leacock and M. Chodorow, "Combining Local Context and Wordnet Similarity for Word Sense Disambiguation, " WordNet: An Electronic Lexical Database, vol. 49, pp. 265-283, MIT Press, 1998.

[39]. J. Jiang and D. Conrath, " Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy," Proc. Int'l Conf. Research in Computational Linguistics (ROCLING X), 1997.

[40]. M. Jarmasz, "Roget's Thesaurus as a Lexical Resource for Natural Language Processing," technical report, Univ. of Ottowa, 2003.

[41]. V. Schickel-Zuber and B. Faltings, "OSS: A Semantic Similarity Function Based on Hierarchical Ontologies, " proc. Int'l Joint Conf. Artificial Intelligence (I] CAI '07), pp. 551-556, 2007.

[42]. E. Agirre E. Alfonseca, K. Hall, J. Kravalova, M. Pasca, and A. Soroa, " A Study on Similarity and Relatedness Using Distributionsl and Wordnet-Based Approaches," proc. Human Language Technologies: The 2009 Ann. Conf. North Am. Chapter of the Assoc. for Computationsl Linguistics (NAACL-HLT '09) 2009.

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

[44]. T. Hughes and D. Ramage, " Lexical Semantic Relatedness with Random Graph Walks, " proc. Joint Cond. Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-Conll '07) pp. 581-589, 2007.

[45]. E. Gabrilovich and S. Markovitch, "Computing Semantic Relatedness Using Wikipedia-Based Explicit Semantic Analysis, " Proc. Int'l Joint Conf. Artificial Intelligence (I]CAI'07), PP. 1606-1611, 2007.

[46]. Y. Matsuo, J. Mori, M.Hamasaki, K. Ishida, T. Nishimura, H. Takeda, K. H asida, and M. Ishizuka, "Polyphonet: An Advanced Social Network Extraction System, " Proc. 15th Int'l World Wide Web Conf., 2006.

[47]. 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), pp. 79-85, 1998.

Authors Profile

ChandraKala K received B.Tech degree from the College of Engineering, Anantapur, JNTU in 2007. She is currently doing M.Tech [Computer Science Engineering] in UCEK, JNTUKAKINADA, KAKINADA.

Dr.E.V.Prasad is currently working as a Profeesor in Computer Science Engineering, University College of Engineering Kakinada, J N T U KAKINADA, Kakinada. He is currently Registrar of JNTUKAKINADA. He has published so many papers in reputed National and International Journals. His research interests are Web Mining, Network Security and Artificial Intelligence.