An Integrated Approach for Measuring Semantic Similarity between Words and Sentences using Web Search Engine

Kavitha Adhikesavan
Manonmaniam Sundaranor University, India

Abstract: *Semantic similarity measures play vital roles in Information Retrieval (IR) and Natural Language Processing (NLP). Despite the usefulness of semantic similarity measures in various applications, strongly measuring semantic similarity between two words remains a challenging task. Here, three semantic similarity measures have been proposed, that uses the information available on the web to measure similarity between words and sentences. The proposed method exploits page counts and text snippets returned by a web search engine. We develop indirect associations of words, in addition to direct for estimating their similarity. Evaluation results on different data sets shows that our methods outperform several competing methods.*

Keywords: *Semantic similarity, web search engine, higher order association mining, support vector machine.*

Received October 29, 2012; accepted February 27, 2013; published online September 4, 2014

1. Introduction

Similarity is a complex concept which has been widely discussed in the linguistic, philosophical and theory communities [3]. An effective method to compute the similarity between short texts or sentences has many applications in Natural Language Processing (NLP) and related areas such as Information Retrieval (IR) and text filtering. For example, in web page retrieval, text similarity has proven to be one of the best techniques for improving retrieval effectiveness and in image retrieval from the web [3] the use of short text surrounding the images can achieve a higher retrieval precision than the use of the whole document in which the image is embedded. The use of text similarity is valuable for relevance feedback and text categorization, text summarization, word sense disambiguation, methods for automatic evaluation of machine translation, evaluation of text coherence and schema matching in databases.

One of the major drawbacks of most of the existing methods is the domain dependency: Once, the similarity method has designed for a specific application domain, it cannot be adapted easily to other domains.

To overcome this drawback, we propose web-based semantic similarity measure that is fully automatic and independent of the domain in applications requiring small text or sentence similarity measure. The computing of text similarity can be viewed as generic component for the research community dealing with text-related knowledge representation and discovery. web-based similarity measures can be broadly divided into three categories such as first one, measures that rely only on the number of the returned hits, second the measures that download a number of the top ranked documents and then apply text processing techniques and finally measures that combine both approaches.

We have used three approaches: First, higher order association mining has been proposed to acquire similarity between words and sentences. Second, measure that uses classification approach to robustly calculate semantic similarity between two given words or sentences. Third, measuring semantic similarity between words and sentences using Clustering Approach.

2. Related Works

Semantic similarity measures have been used in semantic web related applications such as automatic annotation of web pages, community mining and keyword extraction for inter-entity relation representation. There is an extensive literature on measuring the similarity between long texts or documents [1, 4, 10] but there is less work related to the measurement of similarity between sentences or short texts. Related work can roughly be classified into four major categories: Word co-occurrence/vector-based document model methods, corpus based methods, hybrid methods and descriptive feature information based methods.

The vector based document model methods are commonly used in IR systems, where the document most relevant to an input query is determined by representing a document as a word vector and then queries are matched to similar documents in the document database via a similarity metric [14]. The Latent Semantic Analysis (LSA) [4, 7] and the Hyperspace Analogues to Language (HAL) model [2] are two well-known methods in corpus-based similarity. LSA analyses a large corpus of natural

language text and generates a representation that captures the similarity of words and text passages [5]. The dimension of the word by context matrix is limited to several hundreds because of the computational limit of Singular Value Decomposition (SVD). As a result the vector is fixed and the representation of a short text is very sparse. The HAL method uses lexical co-occurrence to produce a high-dimensional semantic space. The author's experimental results showed that HAL was not as promising as LSA in the computation of similarity for short texts.

Hybrid methods have been used for both corpus-based measures [16] and knowledge-based measures [18] of word semantic similarity to determine the text similarity. Mihalcea *et al*. [11] suggest a combined method for measuring the semantic similarity of texts by exploiting the information that can be drawn from the similarity of the component words. Specifically, they use two corpus-based measures, Point wise Mutual Information and IR (PMI-IR) and LSA [5] and six knowledge-based measures [3, 6, 7, 9, 13, 17] of word semantic similarity and combine the results to show how these measures can be used to derive a text-to-text similarity metric. They evaluate their method on a paraphrase recognition task. The main drawback of this method is that it computes the similarity of words from eight different methods, which is not computationally efficient.

Li *et al.* [8] proposed another hybrid method that derives text similarity from semantic and syntactic information contained in the compared texts. Their proposed method dynamically forms a joint word set only using all the distinct words in the pairs of sentences. For each sentence, a raw semantic vector is derived with the assistance of the word net lexical database [12]. A word order vector is formed for each sentence, again using information from the lexical database. Since, each word in a sentence contributes differently to the meaning of the whole sentence, the significance of a word is weighted by using information content derived from a corpus. By combining the raw semantic vector with information content from the corpus, a semantic vector is obtained for each of the two sentences. Semantic similarity is computed based on the two semantic vectors. An order similarity is calculated using the two order vectors. Finally, combining semantic similarity and order similarity derives the sentence similarity.

Feature-based methods try to represent a sentence using a set of predefined features. Similarity between two texts is obtained through a trained classifier. But, finding effective features and obtaining values for these features from sentences make this category of methods more impractical. Sahami *et al*. 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 are not compared their similarity measure with taxonomy-based similarity measures.

Leacock proposed a model combining local context and Word Net similarity for word sense identification [6]. Another approach of double-checking model using text snippets returned by a web search engine to compute semantic similarity between words. For two words P and Q, they collect snippets for each word from a web search engine. Then, they count the occurrences of word P in the snippets for word Q and the occurrences of word Q in the snippets for word P. These values are combined nonlinearly to compute the similarity between P and Q. This method depends heavily on the search engine's ranking algorithm. Although, two words P and Q might be very similar, there is no reason to believe that one can find Q in the snippets for P, or vice versa. This observation is confirmed by the experimental results in their paper which reports zero similarity scores for many pairs of words in the Miller *et al.* [12] dataset. Siddiqui *et al*. [15] developed a method to retrieve the term which are similar using corpus. A corpus is created by collecting information from internet.

3. Proposed Method

The study of semantic similarity between words has long been an integral part of IR and NLP. Semantic similarity between entities changes over time and across domains. An automatic method has been proposed to measure semantic similarity between words or sentences using web search engines. Because of the vastly numerous documents and the high growth rate of the web, it is difficult to analyse each document separately and directly. Web search engines provide an efficient interface to this vast information.

3.1. Outline

Three semantic similarity measures have been proposed here. First method uses association rule mining to calculate semantic similarity between words and sentences. Second method uses Support Vector Machine (SVM) and integrates both page counts and snippets to measure semantic similarity between a given pair of words and sentences. Third method uses sequential clustering algorithm to measure semantic similarity between words and sentences. An automatic lexico-syntactic pattern extraction algorithm has been explained. The patterns extracted have been ranked by our algorithm according to their ability to express semantic similarity.

3.2. Preprocessing the Documents from Google

The snippets downloaded from Google directly are not possible for there are a lot of semantic-unrelated words and word in different form will bring in negative impact in our calculation. Therefore, the following steps are to agreement with the snippets:

1. Delete stop words. Words like 'a', 'the', 'of' and so on called stop words are meaningless for semantic analysis.
2. Use statistical work on the snippets from Google, words in different form will bring in disadvantage influence. Some sort of an algorithm has to be used to void it. Stemmer Algorithm 3 gives us critical help to deal the text.

3.3. About Term Frequency-Inverse Document Frequency (TF-IDF)

The TF-IDF is a weight often used in IR and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

4. Using Association Rule Mining

4.1. Higher Order Association Mining

A novel approach called higher order association mining is proposed to mine word similarity. Exploit indirect associations of words, in addition to direct ones for estimating their similarity. If word *A* co-occurs with word *B*, we say *A* and *B* share a first order association between them. If *A* co-occurs with *B* in some documents and *B* with *C* in some others, then *A* and *C* are said to share a second order co-occurrence via *B*. Higher orders of co-occurrence may similarly be defined as in Figure 1. An algorithm has been presented for mining higher order co-occurrences. A weighted linear model is used to combine the contribution of these higher orders into a word similarity model. This algorithm is used for mining higher order associations between words. The strengths of these associations are combined to yield an estimate of word similarity.

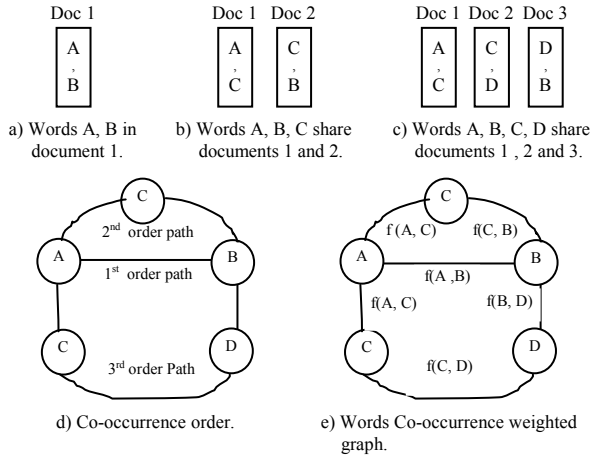

Figure 1. Graphical representation of higher order co-occurrences.

The applicability of matrix operations is explored to directly compute the strengths of higher order associations. First approach, start by computing a first order co-occurrence matrix. For $|W|$ words in the feature set, this is a $|W|\times|W|$ matrix which has a value 1 in the i, j^{th} element if word i co-occurs with word j in at least one document. For all pairs of words that do not co-occur in any document, the corresponding element in the matrix is 0. The diagonal values are set to zero since we are not interested in trivial co-occurrence of a word with itself. The first-order co occurrence matrix is calculated using the following steps:

- *Step* 1: The term document matrix A is multiplied with its transpose A^T to obtain the $|W|\times|W|$ matrix T_0.
- *Step* 2: All non-zero values of T_0 are set to 1 and the diagonal values are set to zero to yield a binary first order co-occurrence matrix T.
- *Step* 3: The second order co-occurrence matrix T_2 *can* be calculated by squaring T. The third order matrix T_2 is given as T_3.

Other higher order co-occurrence matrices can be calculated similarly. Before a matrix is reduced to binary, the value of i, j^{th} element is the number of co-occurrence paths between words i and j. The strength of a first order co-occurrence path is the number of documents in which two words co-occur. The strength of a second order co-occurrence path between words a and b is the number of distinct words c such that a co-occurs with c and b co-occurs with c. Implementing the above algorithm revealed a critical shortcoming d and b. But in addition we also need to ensure that d is not the same as a and c is not the same as b and this is not taken care of. Thus, the strengths of third order associations were over-estimated by the algorithm. To address this limitation a correction is needed on this algorithm. The brute force approach of explicitly counting terms that satisfy the above-mentioned constraint instead of blindly cubing the binary matrix T, turned out to be computationally expensive. A technique has been presented below that rewrites this procedure as an equivalent matrix manipulation, which can be implemented efficiently in matrix processing environments. Let T be the matrix of first order connections with diagonal elements set to zero. For third-order co-occurrences, paths of type *i-j-k-l* for all i and l has been searched to enumerate. Now, Equation 1:

$$(T^3)_{jl} = \sum_{j,k} T_{ij}\, T_{jk}\, T_{kl} \tag{1}$$

Is the total number of such paths, including paths of type *i-j-i-l* and *i-l-k-l*, which wish to exclude. Let n_i be the number of paths of type *i-j-i*. This is equal to the total number of paths originating from i. Evaluate n_i by summing the rows (or columns) of T as in Equation 2.

$$n_i = \sum_j T_{ij} \tag{2}$$

Now, the number of paths of type *i-j-i-l* is $n_i T_{il}$ and for type *i-l-k-l* the count is $n_i T_{il}$. If $T_{il} \neq 0$, the path *i-j-i-j* have been encountered twice, so the total number of invalid paths is $(n_i+n_l-1)T_{ij}$. Equivalently, if construct a discount matrix D whose elements $D_{il}=(n_i+n_1-1)$, then the number of invalid paths between words i and j is

given by the i, j^{th} element of the point wise product $D*T$. The following procedure has been used:

1. Calculate T.
2. Enumerate and discount the invalid paths as above. $T3$- $D*T$ is the revised third order matrix.

4.2. Modelling Word Similarities

Once higher order co-occurrences are mined, we need to translate them into a measure of similarity between words. Intuition suggests that very high order co-occurrences do not really indicate similarity. In a study of higher order associations in the context of Latent Semantic Indexing (LSI), the authors report experimental evidence to confirm that associations beyond an order of 3 have a very weak influence on similarity modelled by LSI. In our word similarity model, which ignore the effects of orders higher than 3.

In the last section, the strength of a higher order association has been used between two terms as the number of co-occurrence paths between those terms. Let *first-order*(a, b), *second-order*(a, b) and *third-order*(a, b) denote the strengths of first, second and third order associations between terms a and b respectively. The similarity between terms a and b can be expressed as a weighted linear combination of the strengths of the first three orders of co-occurrence as in Equation 3.

$$Similarity(a, b) = \alpha\, first\text{-}order(a, b) + \beta\, second\text{-}order(a, b) + \gamma\, third\text{-}order(a, b) \quad (3)$$

Note that, higher the order of association, the larger the number of co-occurrence paths (Since, $T^n_{ij} > T^m_{ij}$, if $n>m$ and if for all $T_{ij} \neq 0$, $T_{ij} \geq 1$, which is true in our case), and hence, the greater the strength of association. Thus, to make α, β and γ comparable to each other, we need to normalize *first-order*(a, b), *second-order*(a, b) and *third-order* (a, b) to values in [0, 1]. In our implementation, this can be achieved by dividing each of these values by the maximum value between any pair of words corresponding to that order.

4.3. Using Classification Approach

The proposed method integrates both page counts and snippets to measure semantic similarity between a given pair of words. Four similarity scores have been defined using page counts. Then, an automatic lexico-syntactic pattern extraction algorithm has been explained. The patterns extracted have been ranked by our algorithm according to their ability to express semantic similarity. Two-class SVMs have been used to find the optimal combination of page counts-based similarity scores and top-ranking patterns. The SVM is trained to classify synonymous word-pairs and non-synonymous word-pairs. Select synonymous word-pairs from Wordnet synsets. Non-synonymous word-pairs are automatically created using a random shuffling technique. Convert the output of SVM into a posterior probability. The semantic similarity between two words has been used as the posterior probability that they belong to the synonymous-words class.

4.4. Page Count-Based Similarity Scores

Page counts for the query A and B, can be considered as an approximation of co-occurrence of two words (or multi word phrases) A and B on the web. However, page counts for the query A and B alone do not accurately express semantic similarity. One must consider the page counts not just for the query A and B, but also for the individual words A and B to assess semantic similarity between A and B. Four popular co-occurrence measures; jaccard, overlap (simpson), dice and Point-wise Mutual Information (PMI) have been modified, to compute semantic similarity using page counts. For the remainder of this paper, the notation $H(A)$ has been used to denote the page counts for the query A in a search engine. The WebJaccard coefficient between words (or multi-word phrases) A and A, WebJaccard (A, B) is defined as in Equation 4.

$$sim(q, d_j) = J(A,B) = \frac{|A \cap B|}{|A \cup B|} \cong \frac{\sum_{k=1}^{n} wkqwkj}{\sum_{k=1}^{n} wkq^2 + \sum_{k=1}^{n} wkj^2 - \sum_{k=1}^{n} wkqwkj} \quad (4)$$

Therein, $A \cap B$ denotes the conjunction query A and B. Given the scale and noise in web data, it is in Equation 5.

$$sim(q, d_j) = J(A,B) = \frac{|A \cap B|}{|A \cup B|} \cong \frac{\sum_{k=1}^{n} wkqwkj}{\sum_{k=1}^{n} wkq^2 + \sum_{k=1}^{n} wkj^2 - \sum_{k=1}^{n} wkqwkj} \quad (5)$$

Possible those two words may appear on some pages purely accidentally. In order to, reduce the adverse effects attributable to random co-occurrences, we set the WebJaccard coefficient to zero if the page count for the query $A \cap B$ is less than a threshold $c5$.

Similarly, we define *Web Overlap*, *Web Overlap* (A, B) as in Equation 6.

$$sim(q, d_j) = O(A,B) = \frac{|A \cap B|}{min(|A|, |B|)} \cong \frac{\sum_{k=1}^{n} wkqwkj}{min(\sum_{k=1}^{n} wkq^2, \sum_{k=1}^{n} wkj^2)} \quad (6)$$

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 (A, B) is defined as in Equation 7.

$$sim(q, d_j) = D(A, B) = \frac{|A \cap B|}{\alpha|A| + (1-\alpha)|B|} \cong \frac{\sum_{k=1}^{n} wkqwkj}{\alpha \sum_{k=1}^{n} wkq^2 + (1-\alpha)} (\alpha \in [0,1]) \quad (7)$$

We define *WebPMI* as a variant form of PMI using page counts in Equation 8.

$$WebPMI(A, B) = \begin{cases} 0 & if\ H(A \cap B) \leq c \\ \log 2 \left(\frac{\frac{H(A \cap B)}{N}}{\frac{H(A) H(B)}{N}} \right) & otherwise \end{cases} \quad (8)$$

Here, N is the number of documents indexed by the search engine. Probabilities in Equation 5 are estimated

according to the maximum likelihood principle. To calculate PMI accurately using Equation 5, N must 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.

4.5. Extracting Patterns from Snippets

Text snippets are returned by search engines alongside with the search results. They provide valuable information regarding the local context of a word. lexico-syntactic patterns has been extracted to indicate various aspects of semantic similarity. Our pattern extraction algorithm is illustrated below.

Algorithm 1: Extracting patterns from snippets.

Comment: *Given a set S of word-pairs, extract patterns.*
for each word-pair(*A*, *B*) *Є S*
  do D ←*GetSnippets*("*A B*")
  N ←*null*
  for each snippet dЄ D
    do N ← *N GetNgrams*(*d*, *A*, *B*)
    Pats ← *CountFreq*(*N*)
return (Pats)

Given a set S of synonymous word-pairs, *GetSnippets* function returns a list of text snippets for the query "*A*" and "*B*" for each word-pair *A*, *B* in *S*. For each snippet found, the two words in the query are replaced by two wildcards. Let us assume these wildcards to be *X* and *Y*. For each snippet *d* in the set of snippets *D* returned by *GetSnippets*, function *GetNgrams* extract word n-grams for n= 2, 3, 4 and 5. N-grams is selected which contain exactly one *X* and one *Y*. Finally, function Count Freq counts the frequency of each pattern has been extracted. The procedure described above yields a set of patterns with their frequencies in text snippets obtained from a search engine. It considers the words that fall between *X* and *Y* as well as words that precede *X* and succeeds *Y*.

Algorithm 2: GetFeatureVector (A, B).

Comment: *Given a word-pair A, B get its feature vector F.*
D ← *GetSnippets* ("*A B*")
N ← *null*
for each snippet d Є D
  do N ← *N* + *GetNgrams*(*d*, *A*, *B*)
  SelPats ← *SelectPatterns* (*N*, *GoodPats*)
  PF ←*Normalize* (*SelPats*)
  F←[*PF*, *WebJaccard*, *WebOverlap*, *WebDice*, *WebPMI*]
return (*F*)

For each pair of words (*A*, *B*), a feature vector *F* has been created as shown in Figure 4. First, query Google for "*A*" and "*B*" and collect snippets. Then replace the query words *A* and *B* with two wildcards *X* and *Y*, respectively in each snippet. Function *GetNgrams* extracts n-grams for n= 2, 3, 4 and 5 from the snippets. Select n-grams having exactly one *X* and one *Y* are used in the pattern extraction algorithm in Figure 3. Assume the set of patterns selected based on their X^2 values in section 3.2 to be *GoodPats*. Then, the function Select Patterns selects the n-grams from *N* which appear in *GoodPats*. In *normalize*(*SelPats*), normalize the count of each pat-tern by diving it from the total number of counts of the observed patterns. This function returns a vector of patterns where each element is the normalized frequency of the corresponding pattern in the snippets for the query "*A*" "*B*". Append similarity scores calculated using page counts in section 3.2 to create the final feature vector F for the word-pair (*A*, *B*). This procedure yields a 204 dimensional feature vector *F*. Feature vectors are formed for all synonymous word-pairs as well as for non-synonymous word-pairs. A two-class SVM is trained with the labelled feature vectors. Once SVM using synonymous and non- synonymous word pairs have been trained, it can be used to compute the semantic similarity between two given words. The same method has been used to generate feature vectors for training, feature vector *F* has been created for the given pair of words (*A*', *B*'), between which need to measure the semantic similarity. The semantic similarity SemSim (*A*', *B*') between *A*' and *B*' has been defined as the posterior probability *Prob*(*F*'| *synonymous*) that feature vector *F*' belongs to the synonymous-words (positive) class.

$$SemSim(A', B') = Prob(F' | synonymous)$$

Being a large-margin classifier, the output of an SVM is the distance from the decision hyper-plane. However, this is not a calibrated posterior probability. Sigmoid function has been used to convert this uncalibrated distance into a calibrated posterior probability. Being a large-margin classifier, the output of an SVM is the distance from the decision hyper-plane. However, this is not a calibrated posterior probability. Sigmoid functions have been used to convert this uncalibrated distance into a calibrated posterior probability.

4.6. Using Clustering Approach

4.6.1. Semantic Similarity between Words

In this approach, given two words, the semantic similarity between two words is calculated using sequential clustering algorithm. To represent the numerous semantic relations that exist between two words lexical patterns are extracted from snippets retrieved from a web search engine. Then, the extracted patterns are clustered to identify the semantically related patterns. Using the pattern clusters a feature vector is defined to represent two words and the semantic similarity is computed by taking into account the inter-cluster correlation.

4.6.2. Extracting Lexical Patterns

The relational model is used to compute semantic similarity between two words, the numerous lexical

patterns are extracted from contexts in which those two words appear. For this purpose, a pattern extraction algorithm is proposed using snippets retrieved from a web search engine. The proposed method requires no language-dependent preprocessing such as part-of-speech tagging or dependency parsing, which can be both time consuming at web scale, and likely to produce incorrect results because of the fragmented and ill-formed snippets. Given two words a and b, query a web search engine to download the snippets. For a snippet S, retrieved for a word pair (a, b), the two words a and b are replaced with two variables X and Y respectively. All numeric values are replaced by D, a marker for digits. All subsequences of words are generated from S that satisfies 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 have gaps.
4. All negation contractions must be expanded in a context.

4.6.3. Clustering Lexical Patterns

A semantic relation can be expressed using more than one pattern. By grouping the semantically related patterns, the model complexity in relational model can be reduced. The distributional hypothesis is used to find semantically related lexical patterns. The distributional hypothesis states that words that occur in the same context have similar meanings. If two lexical patterns are similarly distributed over a set of word pairs, then from the distributional hypothesis it follows that the two patterns must be similar. A pattern p is represented by a vector p in which the i^{th} element is the frequency $f(a_i, b_i, p)$ of p in a word pair (a_i, b_i). Given a set P of patterns and a similarity threshold μ, the sequential clustering algorithm returns clusters of similar patterns. The patterns are sorted in the descending order of their total occurrences in all word pairs. The total occurrences of a pattern p is defined as $\mu(p)$ in Equation 9.

$$\mu(p) = \sum_{(a,b) \in w} f(a, b, p) \quad (9)$$

Here, W is the set of word pairs. Then, a pattern p_i is taken from the ordered set P and finds the cluster, $c^* \in C$ that is most similar to p_i. Similarity between p_i and the cluster centroid c_j is computed using cosine similarity. The centroid vector c_j of cluster c_j is defined as the vector sum of all pattern vectors for patterns in that cluster as in Equation 10.

$$c_j = \sum_{p \in c_j} p \quad (10)$$

If the maximum similarity exceeds the threshold μ, p_i is appended to C *Otherwise, a new cluster is formed and $\{p_i\}$ is appended to it. After all patterns are clustered, the (i, j) element of the inter-cluster correlation matrix Λ is computed as the inner product between the centroid vectors c_i and c_j of the corresponding clusters i and j. The parameter μ determines the purity of the formed clusters moreover, sorting the patterns by their total word pair frequency prior to clustering ensures that the final set of cluster contains the most common relations in the dataset. The sequential clustering algorithm is used to cluster the extracted patterns. The parameter μ is set as follows. The value of theta is varied from 0 to 1 and used to cluster the extracted set of patterns. The resultant set of clusters is used to represent a word pair by a feature vector.

4.6.4. Computation of Semantic Similarity

The semantic similarity is computer using the Equation 11.

$$Sim(a, b) = X^T_{ab}\, \sigma \Lambda \quad (11)$$

Where X^T_{ab} is a feature vector representing the words a and b. The vector σ represents synonymous word pairs and Λ is the inter-cluster correlation matrix.

4.6.5. Semantic Similarity between Sentences

The steps for computing semantic similarity between two sentences:

- Each sentence is partitioned into a list of tokens.
- Part-of-speech disambiguation (or tagging).
- Stemming words.
- Determine the most appropriate sense for every word in a sentence.
- Compute the similarity of the sentences based on the similarity of the pairs of words.

These steps are the same as described earlier in classification approach. Finally the similarity of the sentences if calculated based on the similarity of the pair of words and the similarity of these word pair is calculated using sequential clustering algorithm as described in the earlier section.

5. Experimental Evaluation

The proposed semantic similarity measure has been computed by comparing the similarity scores produced by the proposed measure against Miller-Charles benchmark dataset in Tables 1 and 2.

Table 1. Shows comparison of existing methods with the proposed method.

Similarity Measure	Correlation
Jiang and Conrath	0.695
Hirst St.onge	0.689
Leacock Chodorow	0.821
Lin	0.823
Resnik	0.775
Wu and Palmer	0.803
Our Similarity Measure1	0.831
Our Similarity Measure2	0.812
Our Similarity Measure3	0.772

Table 2. Human and computer rating of the miller-charles dataset.

Word Pair	Web Jaccard	Web Dice	Sahami	Proposed SemSim
Cord-Smile	0.102	0.108	0.09	0
Rooster-Voyage	0.011	0.012	0.197	0.017
Noon-String	0.126	0.133	0.082	0.018
Glass-Magician	0.117	0.124	0.143	0.18
Monk-Slave	0.181	0.191	0.095	0.375
Coast-Forest	0.862	0.87	0.248	0.405
Monk-Oracle	0.016	0.017	0.045	0.328
Lad-Wizard	0.072	0.077	0.149	0.22
Forest-Graveyard	0.068	0.072	0	0.547
Food-Rooster	0.012	0.013	0.075	0.06
Coast-Hill	0.963	0.965	0.293	0.874
Car-Journey	0.444	0.46	0.189	0.286
Crane-Implement	0.071	0.076	0.152	0.133
Brother-Lad	0.189	0.199	0.236	0.344
Bird-Crane	0.235	0.247	0.223	0.879
Bird-Cock	0.153	0.162	0.058	0.593
Food-Fruit	0.753	0.765	0.181	0.998
Brother-Monk	0.261	0.274	0.267	0.377
Asylum-Madhouse	0.024	0.025	0.212	0.773
Furnace-Stove	0.401	0.417	0.31	0.889
Magician-Wizard	0.295	0.309	0.233	1
Journey-Voyage	0.415	0.431	0.524	0.996
Coast-Shore	0.786	0.796	0.381	0.945
Implement-Tool	1	1	0.419	0.684
Boy-Lad	0.186	0.196	0.471	0.974
Automobile-Car	0.654	0.668	1	0.98
Midday-Noon	0.106	0.112	0.289	0.819
Gem-Jewel	0.295	0.309	0.211	0.686
Correlation	0.259	0.267	0.579	0.812

5.1. The Benchmark Dataset

The proposed method has been evaluated against Miller-Charles dataset, a dataset of 30 word-pairs rated by a group of 38 human subjects. The word pairs are rated on a scale from 0 (no similarity) to 4 (perfect synonymy). Miller-Charles data set is a subset of Rubenstein-Goodenough's original data set of 65 word pairs. Although, Miller-Charles experiment was carried out 25 years later than Rubenstein Goodenough's, two sets of ratings are highly correlated. Table 1 presents a comparison of the proposed method to the Wordnet-based methods. The proposed method out performs simple Wordnet-based approaches such as Edge counting and Information Content measures. Unlike the Wordnet based methods, proposed method requires no a hierarchical taxonomy of concepts or sense-tagged definitions of words. Semantic similarity using association, classification and clustering is shown in Figures 2, 3 and 4.

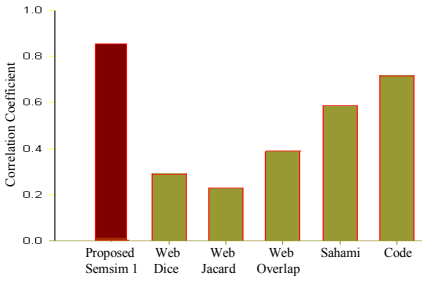

Figure 2. Semantic similarity using association rule mining.

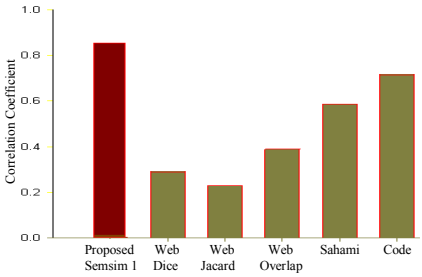

Figure 3. Semantic similarity using classification.

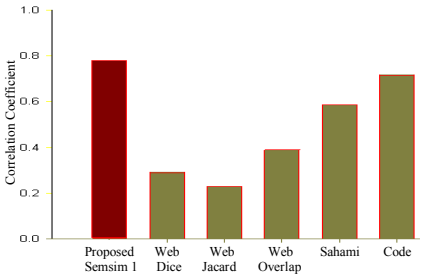

Figure 4. Semantic similarity using clustering.

These results are significant because they are based on a very simple algorithm that relies on assigning relatedness scores to the senses of a target word and the senses of its immediately adjacent neighbours. While the disambiguation results could be improved via the combination of various techniques, our focus is on developing the proposed similarity measure of relatedness as a general tool for NLP and artificial intelligence.

6. Conclusions

The proposed method has been evaluated using Miller-Charles dataset, a dataset of 30 word-pairs rated by a group of 38 human subjects. A high correlation with human ratings was found for semantic similarity on this benchmark dataset. The proposed method has been evaluated with human judgments and found it to be reasonably correlated. These results are significant because they are based on a very simple algorithm that relies on assigning relatedness scores to the senses of a target word and the senses of its immediately adjacent neighbours. While the disambiguation results could be improved via the combination of various techniques, our focus is on developing the proposed similarity measure of relatedness as a general tool for NLP and artificial intelligence.

References

[1] Aguitman A., Menczer F., Roinestad H., and Vespignani A., "Algorithmic Detection of Semantic Similarity," *in Proceedings of the 14th International Conference on World Wide Web*, pp. 107-116, Chiba, Japan, 2005

[2] Burgess K. and Lund K., "Explorations in Context Space: Words, Sentences, Discourse," *Discourse Processes*, vol. 25, no. 2-3, pp. 211-257, 1998.

[3] Jiang J. and Conrath D., "Semantic Similarity based on Corpus Statistics and Lexical Taxonomy," *in Proceedings of the International Conference Research on Computational Linguistics*, Taiwan, pp. 19-33, 1997.
[4] Landauer K. and Dumais T., "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of the Acquisition, Induction and Representation of Knowledge," *Psychological Review*, vol. 104, no. 2, pp. 211-240, 1997.
[5] Landauer K., Foltz W., and Laham D., "Introduction to Latent Semantic Analysis," *Discourse Processes*, vol. 25, no. 2, pp. 259-284, 1998.
[6] Leacock C. and Chodorow M., "Combining Local Context and WordNet Sense Similarity for Word Sense Identification," *Word Net: An Electronic Lexical Database*, MIT Press, Cambridge, USA, 1998.
[7] Lesk M., "Automatic Sense Disambiguation using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone," *in Proceedings of the 5th Annual International Conference on Systems Documentation*, New York, USA, pp. 24-26, 1986.
[8] Li Y., McLean D., Bandar Z., O'Shea J., and Crockett K., "Sentence Similarity based on Semantic Nets and Corpus Statistics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, pp. 1138-1149, 2006.
[9] Lin D., "An Information-theoretic Definition of Similarity," *in Proceedings of the 5th International Conference Machine Learning*, Wisconsin, USA, pp. 296-304, 1998.
[10] Meadow T., Boyce R., and Kraft H., *Text Information Retrieval Systems*, Academic Press, California, USA, 2000.
[11] Mihalcea R., Corley C., and Strapparava C., "Corpus-based and Knowledge-based Measures of Text Semantic Similarity," *in Proceedings of the 21st Conference of American Association for Artificial Intelligence*, Massachusetts, USA, pp. 775-780, 2006.
[12] Miller G., Beckwith R., Fellbaum C., Gross D., and Miller K., available at: http://wordnetcode.princeton.edu/5papers.pdf, last visited 2012.
[13] Resnik P., "Using Information Content to Evaluate Semantic Similarity in a Taxonomy," *in Proceedings of the 14th International Conference on Artificial Intelligence*, Montreal, Canada, pp. 448-453, 1995.
[14] Salton G. and Lesk M., "Computer Evaluation of Indexing and Text Processing," *Prentice Hall*, New Jersey, USA, pp. 143-180, 1971.
[15] Siddiqui M., Fayoumi M., and Yusuf N., "A Corpus based Approach to Find Similar Keywords for Search Engine Marketing," *the International Arab Journal of Information Technology*, vol. 10, no. 5, pp. 460-466, 2013.
[16] Turney P., "Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL," *in Proceedings of the 12th European Conference on Machine Learning*, Freiburg, Germany, pp. 491-502, 2001.
[17] Wu Z. and Palmer M., "Verb Semantics and Lexical Selection," *in Proceedings of the 32nd Annual Meeting Association for Computational Linguistics*, New Mexico, USA, pp. 133-138, 1994.

Kavitha Adhikesavan has completed her MCA degree from University of Madras. She is pursuing her PhD degree in MS University, Tirunelveli. PhD degree in computer science for her research in data mining. she is the assistant professor of the department of computer applications, RMK engineering college, India.