Semantic Similarity of Short Texts

Aminul Islam and Diana Inkpen
University of Ottawa
School of Information Tech. and Eng.
Ottawa, Ontario, Canada, K1N 6N5
{mdislam, diana}@site.uottawa.ca

Abstract

This paper presents a method for measuring the semantic similarity of texts using a corpus based measure of semantic word similarity and a normalized and modified versions of the Longest Common Subsequence (LCS) string matching algorithm. Existing methods for computing text similarity have focused mainly on either large documents or individual words. In this paper, we focus on computing the similarity between two sentence or between two short paragraphs. The proposed method can be exploited in a variety of applications involving textual knowledge representation and knowledge discovery. Evaluation results on two different data sets show that our method outperforms several competing methods.

Keywords

Semantic similarity of words, similarity of short texts, corpus-based measures.

1. Introduction

Similarity is a complex concept which has been widely discussed in the linguistic, philosophical, and information theory communities. Frawley [9] discusses all semantic typing in terms of two mechanisms: the detection of similarities and differences. For our task, given two input text segments, we want to automatically determine a score that indicates their similarity at *semantic* level, thus going beyond the simple lexical matching methods traditionally used for this task.

An effective method to compute the similarity between short texts or sentences has many applications in natural language processing and related areas such as information retrieval and text filtering. For example, in web page retrieval, text similarity has proven to be one of the best techniques for improving retrieval effectiveness [33] and in image retrieval from the Web, 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 [3]. The use of text similarity is beneficial for relevance feedback and text categorization [13], [24], text summarization [7], [22], word sense disambiguation [19], methods for automatic evaluation of machine translation [25], [31], evaluation of text coherence [17], and schema matching in databases [26].

One of the major drawbacks of most of the existing methods is the domain dependency: once the similarity method is designed for a specific application domain, it cannot be adapted easily to other domains. To address this drawback, we aim to develop a method 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 a generic component for the research community dealing with text-related knowledge representation and discovery.

This paper is organized as follow: Section 2 presents a brief overview of the related work. Our proposed method is described in Section 3. Evaluation and experimental results are discussed in Section 4.

2. Related Work

There is extensive literature on measuring the similarity between long texts or documents [15], [27], [28], but there is less work related to the measurement of similarity between sentences or short texts [8]. 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-based methods.

The vector-based document model methods are commonly used in Information Retrieval (IR) systems [28], 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 [37].

The Latent Semantic Analysis (LSA) [15], [16] and the Hyperspace Analogues to Language (HAL) model [2] are two well known methods in corpus-based similarity. LSA analyzes a large corpus of natural language text and generates a representation that captures the similarity of words and text passages. 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 authors' experimental results showed that HAL was not as promising as LSA in the computation of similarity for short texts.

Hybrid methods use both corpus-based measures [38] and knowledge-based measures [18] of word semantic similarity to determine the text similarity. Mihalcea et al. [30] 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, PMI-IR (Pointwise Mutual Information and Information Retrieval) [38] and LSA (Latent Semantic Analysis) [16] and six knowledge-based measures [12], [18], [19], [23],

[34], [39] 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. [20] propose 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 WordNet lexical database [32]. 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, the sentence similarity is derived by combining semantic similarity and order 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.

3. Proposed Method

The proposed method derives text similarity of two texts by combining semantic similarity and string similarity, with normalization. We call our proposed method the Semantic Text Similarity (STS) method. We investigate the importance of including string similarity by a simple example. Let us consider a pair of texts, T_1 and T_2 that contain a *proper noun* (*proper name*) 'Maradona' in T_1. In T_2 the name 'Maradona' is misspelled to 'Maradena'.

T_1 : Many consider Maradona as the best player in soccer history.

T_2 : Maradena is one of the best soccer players.

Dictionary-based similarity measure can not provide any similarity value between these two proper names. And the chance to obtain a similarity value using corpus-based similarity measures is very low. We obtain a good similarity score if we use string similarity measures. The following sections present a detailed description of each of the above mentioned functions.

3.1 String Similarity between Words

We use the *longest common subsequence* (LCS) [1], [14] measure with some normalization and small modifications for our string similarity measure. We use three different modified versions of LCS and then take a weighted sum of these[1]. Melamed [29] normalized LCS by dividing the length of the longest common subsequence by the length of the longer string and called it *longest common subsequence ratio* (LCSR). But LCSR does not take into account of the length of the shorter string which sometimes has a significant impact on the similarity score.

We normalize the *longest common subsequence* (LCS) so that it takes into account of the length of both the shorter and the longer string and call it *normalized longest common subsequence* (NLCS) which is,

$$v_1 = NLCS(r_i, s_j) = \frac{\{length(LCS(r_i, s_j))\}^2}{length(r_i) \times length(s_j)} \quad (1)$$

While in classical LCS, the common subsequence needs not be consecutive, in text matching, consecutive common subsequence is important for a high degree of matching. We use *maximal consecutive longest common subsequence* starting at character 1, $v_2 = MCLCS_1$ (Fig. 1) and *maximal consecutive longest common subsequence* starting at any character n, $v_3 = MCLCS_n$ (Fig. 2). In Fig. 1, we present an algorithm that takes two strings as input and returns the shorter string or maximal consecutive portions of the shorter string that consecutively match with the longer string, where matching must be from first character (character 1) for both strings. In Fig. 2, we present another algorithm where matching may start from any character (character n). We also normalize $MCLCS_1$ and $MCLCS_n$.

We take the weighted sum of the values v_1, v_2, and v_3 to determine string similarity score, where w_1, w_2, w_3 are weights and $w_1+w_2+w_3=1$. Therefore, the similarity of the two strings is: $\alpha = w_1v_1 + w_2v_2 + w_3v_3$ (2)

We set equal weights for our experiments. [2]

Algorithm $MCLCS_1$
Input: r_i, s_j // r_i and s_j are two input strings where
// $|r_i| = \tau$, $|s_j| = \eta$ and $\tau \leq \eta$ as mentioned earlier.
1. $\tau \leftarrow |r_i|$, $\eta \leftarrow |s_j|$
2. **while** $|r_i| > 0$
3. **if** $r_i \subset s_j$ // i.e., $s_j \cap r_i = r_i$
4. **return** r_i
5. **else** $r_i \leftarrow r_i \setminus c_\tau$ // i.e., remove the right-most character from r_i
6. **end if**
7. **end while**
Output: r_i // r_i is the Maximal Consecutive LCS starting at character 1

Fig. 1. Maximal consecutive LCS starting at character 1.

[1] We use modified versions because in our experiments we obtained better results (precision and recall) for text matching on a sample of data than when using the original LCS, or other string similarity measures.

[2] We use equal weights in several places in this paper in order to keep the system unsupervised. If development data would be available, we could adjust the weights.

Algorithm *MCLCS$_n$*

Input: r_i, s_j // r_i and s_j are two input strings
// where $|r_i| = \tau$, $|s_j| = \eta$ and $\tau \leq \eta$.

1. **while** $|r_i| > 0$
2. determine all n-grams from r_i where $n = 1 .. |r_i|$ and $\overline{r_i}$ is the set of n-grams
3. **if** $x \in S_j$ where $\{x \mid x \in \overline{r_i}, x = \textbf{Max}(\overline{r_i})\}$
// i is the number of n-grams and Max ($\overline{r_i}$)
// returns the maximum length n-gram from $\overline{r_i}$
4. **return** x
5. **else** $\overline{r_i} \leftarrow \overline{r_i} \setminus x$ // remove x from set $\overline{r_i}$
6. **end if**
7. **end while**

Output: x // x is the Maximal Consecutive
// LCS starting at any character n

Fig. 2. Maximal consecutive LCS starting at any character *n*

Algorithm *semanticMatching*

Input: r_i, s_j // r_i and s_j are two input words
// where $|r_i| = \tau$, $|s_j| = \eta$ and $\tau \leq \eta$.

1. $v \leftarrow$ ***SOCPMI***(r_i, s_j) // This method determines
// semantic similarity between two words. Any //other similarity method can also be used instead.
2. **if** $v > \lambda$ // λ is the maximum possible similarity values
3. $v \leftarrow 1$
4. **else** $v \leftarrow v / \lambda$
5. **end if**

Output: v // v is the semantic similarity value
// between 0 and 1, inclusively

Fig. 3. Semantic similarity matching.

3.2 Semantic Similarity between Words

There is a relatively large number of word-to-word similarity metrics in the literature, ranging from distance-oriented measures computed on semantic networks or knowledge base (or dictionary/thesaurus-based measures), to metrics based on models of information theory (or corpus-based measures) learned from large text collections. A detailed review on word similarity can be found in [21], [35]. We focus our attention on corpus-based measures because of their large type coverage.

PMI-IR [38] is a simple method for computing corpus-based similarity of words which uses Pointwise Mutual Information. PMI-IR used AltaVista Advanced Search query syntax to calculate the probabilities. LSA, another corpus-based measure, analyzes a large corpus of natural text and generate a representation that captures the similarity of words (discussed in the Related Work section).

We use the Second Order Co-occurrence PMI (SOC-PMI) word similarity method [10] that uses Pointwise Mutual Information to sort lists of important neighbor words of the two target words from a large corpus. The method considers the words which are common in both lists and aggregate their PMI values (from the opposite list) to calculate the relative semantic similarity. We define the *pointwise mutual information* function for only those words having $f^b(t_i, w) > 0$,

$$f^{pmi}(t_i, w) = \log_2 \frac{f^b(t_i, w) \times m}{f^t(t_i) f^t(w)},$$

where $f^t(t_i)$ tells us how many times the type t_i appeared in the entire corpus, $f^b(t_i, w)$ tells us how many times word t_i appeared with word w in a context window words and m is total number of tokens in the corpus. Now, for word w_1, we define a set of words, X, sorted in descending order by their PMI values with w_1 and taken the top-most β_1 words having $f^{pmi}(t_i, w_1) > 0$.

$X = \{X_i\}$, where $i = 1, 2, \ldots, \beta_1$ and

$f^{pmi}(t_1, w_1) \geq f^{pmi}(t_2, w_1) \geq \ldots f^{pmi}(t_{\beta 1\text{-}1}, w_1) \geq f^{pmi}(t_{\beta 1}, w_1)$

Similarly, for word w_2, we define a set of words, Y, sorted in descending order by their PMI values with w_2 and taken the top-most β_2 words having $f^{pmi}(t_i, w_2) > 0$. The value of β (either β_1 or β_2) is related to how many times a word w appears in the corpus, i.e., the frequency of w as well as the number of types in the corpus. Then we define the *β-PMI summation* function. For word w_1, the *β-PMI summation* function is:

$$f^{\beta}(w_1) = \sum_{i=1}^{\beta_1} \left(f^{pmi}(X_i, w_2) \right)^{\gamma},$$

where, $f^{pmi}(X_i, w_2) > 0$ and $f^{pmi}(X_i, w_1) > 0$

which sums all the positive PMI values of words in the set Y also common to the words in the set X. In other words, this function actually aggregates the positive PMI values of all the semantically close words of w_2 which are also common in w_1's list. The higher the value of γ is, the greater emphasis on words having very high PMI values with w_1 is given. Similarly, we calculate the *β-PMI summation* function for word w_2. Finally, we define the *semantic PMI similarity* function between the two words, w_1 and w_2,

$$Sim(w_1, w_2) = \frac{f^{\beta}(w_1)}{\beta_1} + \frac{f^{\beta}(w_2)}{\beta_2}$$

We normalize the semantic word similarity (Fig. 3), so that it provides a similarity score between 0 and 1 inclusively. The word similarity method is a separate module in our Text Similarity Method. Therefore any other word similarity method could be substituted instead of SOC-PMI. In that case, we need to set λ to the maximum similarity value specific to that method.

3.3 Overall Sentence Similarity

Our task is to derive a score between 0 and 1 inclusively that will indicate the similarity between two texts P and R at semantic level. The main idea is to find, for each word in the first sentence, the most similar matching in the second sentence. The method consists in the following six steps:

Step 1: We use all special characters, punctuations, and capital letters, if any, as initial word boundary and eliminate all these special characters, punctuations and stop words. We lemmatize each of the segmented words to generate tokens. After cleaning we assume that the text $P = \{p_1, p_2 \ldots, p_m\}$ has m tokens and the text $R = \{r_1, r_2 \ldots, r_n\}$ has n tokens and $n \geq m$. Otherwise, we switch P and R.

Step 2: We count the number of p_i's (say, δ) for which $p_i = r_j$, for all $p \in P$ and for all $r \in R$. I.e., there are δ tokens in P that exactly match with R, where $\delta \leq m$. We remove all δ tokens from both of P and R. So, $P = \{p_1, p_2 \ldots, p_{m-\delta}\}$ and $R = \{r_1, r_2 \ldots, r_{n-\delta}\}$. If all the terms match, $m-\delta = 0$, we go to step 6.

Step 3: We construct a $(m-\delta)\times(n-\delta)$ *string similarity matrix* (say, $M_1 = (\alpha_{ij})_{(m-\delta)\times(n-\delta)}$) using the following process: we assume any token $p_i \in P$ has τ characters, i.e., $p_i = \{c_1c_2\ldots c_\tau\}$ and any token $r_j \in R$ has η characters, i.e., $r_j = \{c_1c_2 \ldots c_\eta\}$ where $\tau \leq \eta$. In other words, η is the length of the longer token and τ is the length of the shorter token. We calculate the followings:

$v_1 \leftarrow NLCS(p_i, r_j)$,

$v_2 \leftarrow NMCLCS_1(p_i, r_j)$

$v_3 \leftarrow NMCLCS_n(p_i, r_j)$,

$\alpha_{ij} \leftarrow w_1v_1 + w_2v_2 + w_3v_3$

i.e., α_{ij} is a weighted sum of v_1, v_2, and v_3 where w_1, w_2, w_3 are weights and $w_1+w_2+w_3=1$. We set equal weights for our experiments.

We put α_{ij} in row i and column j position of the matrix for all $i = 1 \ldots m-\delta$ and $j = 1 \ldots n-\delta$.

$$M_1 = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{1j} & \alpha_{1(n-\delta)} \\ \alpha_{21} & \alpha_{22} & \alpha_{2j} & \alpha_{2(n-\delta)} \\ \alpha_{i1} & \alpha_{i2} & \alpha_{ij} & \alpha_{i(n-\delta)} \\ \alpha_{(m-\delta)1} & \alpha_{(m-\delta)2} & \alpha_{(m-\delta)j} & \alpha_{(m-\delta)(n-\delta)} \end{bmatrix}$$

Step 4: We construct a $(m-\delta)\times(n-\delta)$ *semantic similarity matrix* (say, $M_2 = (\beta_{ij})_{(m-\delta)\times(n-\delta)}$) using the following process: We put β_{ij} ($\beta_{ij} \leftarrow semanticMatching(p_i, r_j)$ (Fig. 3) in row i and column j position of the matrix for all $i = 1 \ldots m-\delta$ and $j = 1 \ldots n-\delta$.

$$M_2 = \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{1j} & \beta_{1(n-\delta)} \\ \beta_{21} & \beta_{22} & \beta_{2j} & \beta_{2(n-\delta)} \\ \beta_{i1} & \beta_{i2} & \beta_{ij} & \beta_{i(n-\delta)} \\ \beta_{(m-\delta)1} & \beta_{(m-\delta)2} & \beta_{(m-\delta)j} & \beta_{(m-\delta)(n-\delta)} \end{bmatrix}$$

Step 5: We construct another $(m-\delta)\times(n-\delta)$ *joint matrix* (say, $M = (\gamma_{ij})_{(m-\delta)\times(n-\delta)}$) using

$$M \leftarrow \psi M_1 + \varphi M_2 \qquad (3)$$

(i.e., $\gamma_{ij} = \psi\alpha_{ij} + \varphi\beta_{ij}$) where ψ is the *string matching matrix* weight factor. φ is the *semantic similarity matrix* weight factor, and $\psi + \varphi = 1$. We set equal weights for our experiments.

$$M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{1j} & \gamma_{1(n-\delta)} \\ \gamma_{21} & \gamma_{22} & \gamma_{2j} & \gamma_{2(n-\delta)} \\ \gamma_{i1} & \gamma_{i2} & \gamma_{ij} & \gamma_{i(n-\delta)} \\ \gamma_{(m-\delta)1} & \gamma_{(m-\delta)2} & \gamma_{(m-\delta)j} & \gamma_{(m-\delta)(n-\delta)} \end{bmatrix}$$

After constructing the *joint matrix*, M, we find out the maximum-valued matrix-element, γ_{ij}. We add this matrix element to a list (say, ρ and $\rho \leftarrow \rho \cup \gamma_{ij}$) if $\gamma_{ij} > 0$. We remove all the matrix elements of i'th row and j'th column from M. We repeat the finding of the maximum-valued matrix-element, γ_{ij} adding it to ρ and removing all the matrix elements of the corresponding row and column until either $\gamma_{ij} = 0$, or $m-\delta-|\rho| = 0$, or both.

Step 6: We sum up all the elements in a value ρ and add δ to it to get a total score. We multiply this total score by the reciprocal harmonic mean of m and n to obtain a balanced similarity score between 0 and 1, inclusively.

$$S(P,R) = \frac{(\delta + \sum_{i=1}^{|\rho|} \rho_i) \times (m+n)}{2mn} \qquad (4)$$

4. Evaluation and Experimental Results

In order to evaluate our text similarity measure, we use two different data sets: 30 sentence pairs [20] and the Microsoft paraphrase corpus [6].

4.1 Experiment with Human Similarities of Sentence Pairs

We use the same data set as Li et al. [20] (available at http://www.docm.mmu.ac.uk/STAFF/D.McLean/SentenceResults.htm). Li et al. [20] collected human ratings for the similarity of pairs of sentences following existing designs for word similarity measures. The participants consisted of 32 volunteers, all native speakers of English educated to graduate level or above. Li et al. [20] began with the set of 65 noun pairs from Rubenstein and Goodenough [36] and replaced them with their definitions from the Collins Cobuild dictionary [4]. Cobuild dictionary definitions are written in full sentences, using vocabulary and grammatical structures that occur naturally with the word being explained. The participants were asked to complete a questionnaire, rating the similarity of meaning of the sentence pairs on the scale from 0.0 (minimum similarity) to 4.0 (maximum similarity), as in Rubenstein and Goodenough (R&G) [36]. Each sentence pair was presented on a separate sheet. The order of presentation of the sentence pairs was randomized in each questionnaire. The order of the two sentences making up each pair was also randomized. This was to prevent any bias being introduced by order of presentation. Each of the 65 sentence pairs was assigned a semantic similarity score calculated as the mean of the judgments made by the

participants. The distribution of the semantic similarity scores was heavily skewed toward the low similarity end of the scale. A subset of 30 sentence pairs was selected to obtain a more even distribution across the similarity range. This subset contains all of the sentence pairs rated 1.0 to 4.0 and 11 (from a total of 46) sentences rated 0.0 to 0.9 selected at equally spaced intervals from the list. The detailed procedure of this data set preparation is in [20]. Table 1 shows average human similarity scores along with Li et al.'s Similarity Method scores [20] and our proposed Semantic Text Similarity scores. Human similarity scores are provided as the mean score for each pair and have been scaled into the range [0..1].

Table 1. Results on Li et al. sentence data set

R&G No.	R&G word pair in the sentence	Human Sim. (Mean)	Li et al. Sim. Meth.	STS Meth.	R&G No.	R&G word pair in the sentence	Human Sim. (Mean)	Li et al. Sim. Meth.	STS Meth.
1	Cord Smile	0.01	0.33	0.06	51	Glass Tumbler	0.14	0.65	0.28
5	Autograph Shore	0.01	0.29	0.11	52	Grin Smile	0.49	0.49	0.32
9	Asylum Fruit	0.01	0.21	0.07	53	Serf Slave	0.48	0.39	0.44
13	Boy Rooster	0.11	0.53	0.16	54	Journey Voyage	0.36	0.52	0.41
17	Coast Forest	0.13	0.36	0.26	55	Autograph Signature	0.41	0.55	0.19
21	Boy Sage	0.04	0.51	0.16	56	Coast Shore	0.59	0.76	0.47
25	Forest Graveyard	0.07	0.55	0.33	57	Forest Woodland	0.63	0.7	0.26
29	Bird Woodland	0.01	0.33	0.12	58	Implement Tool	0.59	0.75	0.51
33	Hill Woodland	0.15	0.59	0.29	59	Cock Rooster	0.86	1	0.94
37	Magician Oracle	0.13	0.44	0.20	60	Boy Lad	0.58	0.66	0.60
41	Oracle Sage	0.28	0.43	0.09	61	Cushion Pillow	0.52	0.66	0.29
47	Furnace Stove	0.35	0.72	0.30	62	Cemetery Graveyard	0.77	0.73	0.51
48	Magician Wizard	0.36	0.65	0.34	63	Automobil Car	0.56	0.64	0.52
49	Hill Mound	0.29	0.74	0.15	64	Midday Noon	0.96	1	0.93
50	Cord String	0.47	0.68	0.49	65	Gem Jewel	0.65	0.83	0.65

Fig. 4 shows that our proposed Semantic Text Similarity Measure achieves a high Pearson correlation coefficient of 0.853 with the average human similarity ratings, whereas Li et al.'s Similarity Measure [20] achieves 0.816. The improvement we obtained is statistically significant at the 0.05 level[3]. In the human judging experiment of Li et al. [20] the best human participant obtained a correlation of 0.921 with the mean of the participants and the worst participant obtained 0.594.

4.2 Experiment with Microsoft Paraphrase Corpus

We use the semantic text similarity method to automatically identify if two text segments are paraphrases of each other. We use the Microsoft paraphrase corpus [6], consisting of 4,076 training and

[3] We used the test from http://faculty.vassar.edu/lowry/rdiff.html?

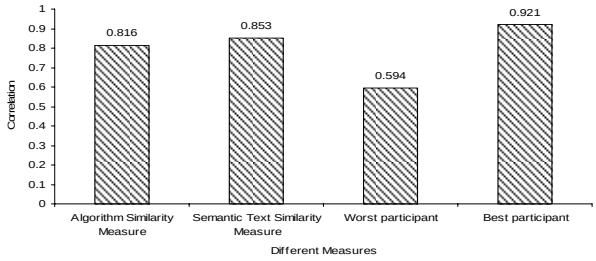

Fig. 4. Similarity correlations.

1,725 test pairs, and determine the number of correctly identified paraphrase pairs in the corpus using the semantic text similarity measure. The paraphrase pairs in this corpus were labeled by two human annotators who determined if the two sentences in a pair were semantically equivalent paraphrases or not. The agreement between the human judges who labeled the candidate paraphrase pairs in this data set was measured at approximately 83%, which can be considered as an upper bound for an automatic paraphrase recognition task performed on this data set.

We acknowledge, as in [5], that the semantic similarity measure for short texts is a necessary step in the paraphrase recognition task, but not always sufficient. There might be cases when the same meaning is expressed in one sentence and the exact opposite meaning in the second sentence (for example by adding the word *not*). For these situations deeper reasoning methods are needed.

We evaluate the results in terms of accuracy, the number of pairs predicted correctly divided by the total number of pairs. We also measure precision ($P = TP / (TP + FP)$), recall ($R = TP / (TP + FN)$) and F-measure ($F = 2PR / (P + R)$). Here, *TP*, *FP* and *FN* stand for True Positive, False Positive and False Negative respectively.

We use eleven different similarity thresholds ranging from 0 to 1 with interval 0.1. In Table 2, when we use similarity threshold score of 1 (i.e., matching word by word exactly, therefore no semantic similarity matching is needed), we obtain recall value of 0.0044 for the test data set. We can consider this score as one of the baselines. Mihalcea et al. [30] mentioned two other baselines: Vector-based and Random. See Table 3 for the results of these baselines and the results of several methods from [30] and [5] (on the test set).

For this paraphrase identification task, we can consider our proposed STS method as a supervised method. Using training data set, we obtain the best accuracy of 72.42% when we use 0.6 as the similarity threshold score. Therefore we can recommend this threshold for use on the test set, achieving an accuracy of 72.64% (our method predicts 1369 pairs as correct, out of which 1022 pairs are correct among the 1725 manually annotated pairs). Our results on the test set are shown in Table 3.

For each candidate paraphrase pair in the test set, we first calculate the semantic text similarity score using (4),

and then label the candidate pair as a paraphrase if the similarity score exceeds a threshold of 0.6. We obtain the same F-measure (81%) at the combined methods from [30] and [5]. We obtain higher accuracy and precision at the cost of decreasing recall.

Table 2. Characteristics of the paraphrase evaluation data set and our results

Number of pairs in (data set)	Number of pairs determined as correct by human annotators (*TP+FN*)	Similarity threshold score in our method	Accuracy (%)	Number of correct pairs (*TP*)	Number of predicted pairs (*TP+FP*)
		0	67.54	2753	4076
		0.1	67.54	2753	4076
		0.2	67.54	2753	4076
		0.3	67.59	2753	4074
		0.4	67.74	2751	4064
4076	2753	0.5	69.53	2708	3905
(Training)		0.6	**72.42**	2435	3241
		0.7	68.45	1874	2281
		0.8	56.67	1085	1183
		0.9	37.78	218	219
		1.0	32.82	15	15
		0	66.49	1147	1725
		0.1	66.49	1147	1725
		0.2	66.49	1147	1725
		0.3	66.49	1147	1725
1725	1147	0.4	66.66	1146	1720
(Test)		0.5	68.86	1128	1646
		0.6	**72.64**	1022	1369
		0.7	68.06	768	940
		0.8	56.29	443	493
		0.9	38.38	86	88
		1.0	33.79	5	5

5. Conclusion

Our proposed STS method achieves a very good Pearson correlation coefficient for 30 sentence pairs data set and outperforms the results obtained by Li et al. [20] (the improvement is statistically significant). For the paraphrase recognition task, our proposed STS method performs similar to the combined unsupervised method [30] and the combined supervised method [5]. The main advantage of our system is that is that it has lower complexity and running time than the other systems [20], [5], [30], because we use only one corpus-based measure, while they combine both corpus-based and WordNet-based measures. For example, Mihalcea et. al [30] use six WordNet-based measures and two corpus-based measures. The complexity of the algorithms and their running time is given mainly by the number of searches in the corpus and in WordNet. We don't use WordNet at all, therefore saving a lot of time. We add the string similarity measure, but this is very fast, because we apply it on short strings (no search needed).

Our method can be used as unsupervised or supervised. For the second task, paraphrase recognition, we used it as supervised, but only to find the best threshold. For the first task, comparing our sentence similarity score to scores assigned by human judges, our system is used as unsupervised (there is no training data available).

Table 3. Text similarity results for paraphrase identification (test set)

Metric	Accuracy	Precision	Recall	F-measure
Semantic similarity (corpus-based)				
PMI-IR	69.9	70.2	95.2	81.0
LSA	68.4	69.7	95.2	80.5
STS	**72.6**	**74.7**	89.1	**81.3**
Semantic similarity (knowledge-based)				
J & C	69.3	72.2	87.1	79.0
L & C	69.5	72.4	87.0	79.0
Lesk	69.3	72.4	86.6	78.9
Lin	69.3	71.6	88.7	79.2
W & P	69.0	70.2	92.1	80.0
Resnik	69.0	69.0	96.4	80.4
Combined(S)	71.5	72.3	92.5	81.2
Combined(U)	70.3	69.6	**97.7**	**81.3**
Baselines				
Threshold-1	33.8	100.0	0.44	0.87
Vector-based	65.4	71.6	79.5	75.3
Random	51.3	68.3	50.0	57.8

6. References

[1] L. Allison, T.I. Dix, "A Bit-String Longest-Common-Subsequence Algorithm," *Information Processing Letters,* vol. 23, pp. 305-310, 1986.

[2] C. Burgess, K. Livesay, and K. Lund, "Explorations in Context Space: Words, Sentences, Discourse," *Discourse Processes*, vol. 25, nos. 2-3, pp. 211-257, 1998.

[3] T.A.S. Coelho, P.P. Calado, L.V. Souza, B. Ribeiro-Neto, and R. Muntz, "Image Retrieval Using Multiple Evidence Ranking," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 4, pp. 408-417, Apr. 2004.

[4] *Collins Cobuild English Dictionary for Advanced Learners*, J. Sinclair, ed., third ed. Harper Collins Pub., 2001.

[5] C. Corley and R. Mihalcea, "Measures of Text Semantic Similarity," *Proc. ACL workshop on Empirical Modeling of Semantic Equivalence*, Ann Arbor, MI, June, 2005.

[6] W. Dolan, C. Quirk, and C. Brockett, "Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources," *Proc. 20th Int'l Conf. Computational Linguistics*, 2004.

[7] G. Erkan and D.R. Radev, "LexRank: Graph-Based Lexical Centrality As Salience in Text Summarization," *J. Artificial Intelligence Research*, vol. 22, pp. 457-479, 2004.

[8] P.W. Foltz, W. Kintsch, and T.K. Landauer, "The Measurement of Textual Coherence with Latent Semantic Analysis," *Discourse Processes*, vol. 25, nos. 2-3, pp. 285-307, 1998.

[9] W. Frawley, *Linguistic Semantics.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1992.

[10] A. Islam and D. Inkpen, "Second Order Co-occurrence PMI for Determining the Semantic Similarity of Words," *Proc. Int'l Conf. on Language Resources and Evaluation*, Genoa, Italy, May, 2006.

[11] M. Jarmasz and S. Szpakowicz, "Roget's Thesaurus and Semantic Similarity," *Proc. Int'l Conf. on Recent Advances in Natural Language Processing*, pp. 212-219, 2003.

[12] J. Jiang and D. Conrath, "Semantic Similarity based on Corpus Statistics and Lexical Taxonomy," *Proc. Int'l Conf. Research in Computational Linguistics*, 1997.

[13] Y. Ko, J. Park, and J. Seo, "Improving Text Categorization Using the Importance of Sentences," *Information Processing and Management*, vol. 40, pp. 65-79, 2004.

[14] G. Kondrak, "N-gram Similarity and Distance," Proc. Twelfth Int'l Conf. on String Processing and Information Retrieval, pp. 115-126, 2005.

[15] T. K. Landauer and S. T. Dumais, "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of the Acquisition, Induction, and Representation of Knowledge," *Psychological Review*, vol. 104, nos. 2 , pp. 211-240, 1997.

[16] T. K. Landauer, P. W. Foltz, and D. Laham, "Introduction to Latent Semantic Analysis," *Discourse Processes*, vol. 25, nos. 2-3, pp. 259-284, 1998.

[17] M. Lapata and R. Barzilay, "Automatic Evaluation of Text Coherence: Models and Representations," *Proc. 19th Int'l Joint Conf. AI*, 2005.

[18] C. Leacock and M. Chodorow, "Combining Local Context and WordNet Sense Similarity for Word Sense Identification," *WordNet, An Electronic Lexical Database,* The MIT Press, 1998.

[19] M. Lesk, "Automatic Sense Disambiguation using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone," *Proc. SIGDOC Conf.*, 1986.

[20] Y. Li, D. McLean, Z. Bandar, J. O'Shea, and K. Crockett, "Sentence Similarity Based on Semantic Nets and Corpus Statistics," *IEEE Trans. Knowledge and Data Eng.*, vol. 18, no. 8, pp. 1138-1149, Aug. 2006.

[21] Y.H. Li, Z. Bandar, and D. McLean, "An Approach for Measuring Semantic Similarity Using Multiple Information Sources," *IEEE Trans. Knowledge and Data Eng.*, vol. 15, no. 4, pp. 871-882, July/ Aug. 2003.

[22] C. Lin and E. Hovy, "Automatic Evaluation of Summaries using n-gram Co-occurrence Statistics," *Proc. Human Language Technology Conf.*, 2003.

[23] D. Lin, "An Information-theoretic Definition of Similarity," *Proc. Int'l Conf. Machine Learning*, 1998.

[24] T. Liu and J. Guo, "Text Similarity Computing Based on Standard Deviation," *Proc. Int'l Conf. on Intelligent Computing*, D.-S. Huang, X.-P. Zhang and G.-B. Huang, eds., LNCS 3644, Springer, pp. 456-464, 2005

[25] Y. Liu and C.Q. Zong, "Example-Based Chinese-English MT," *Proc. 2004 IEEE Int'l Conf. Systems, Man, and Cybernetics*, vols. 1-7, pp. 6093-6096, 2004.

[26] J. Madhavan, P. Bernstein, A. Doan, and A. Halevy, "Corpus-based Schema Matching," *Int'l Conf. Data Eng.*, 2005.

[27] A. Maguitman, F. Menczer, H. Roinestad, and A. Vespignani, "Algorithmic Detection of Semantic Similarity," *Proc. 14th Int'l World Wide Web Conf.*, May 2005.

[28] C.T. Meadow, B.R. Boyce, and D.H. Kraft, *Text Information Retrieval Systems*, second ed. Academic Press, 2000.

[29] I.D. Melamed, "Bitext Maps and Alignment via Pattern Recognition," *Computational Linguistics*, vol. 25, no. 1, pp. 107–130, 1999.

[30] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and Knowledge-based Measures of Text Semantic Similarity," *Proc. American Association for Artificial Intelligence*, Boston, July, 2006.

[31] G.A. Miller and W.G. Charles, "Contextual Correlates of Semantic Similarity," *Language and Cognitive Processes*, vol. 6, no. 1, pp. 1-28, 1991.

[32] G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller, "Introduction to WordNet: An on-line lexical database," *CSL* 43, Cognitive Science Laboratory, Princeton University, Princeton, NJ, 1993.

[33] E.K. Park, D.Y. Ra, and M.G. Jang, "Techniques for Improving Web Retrieval Effectiveness," *Information Processing and Management*, vol. 41, no. 5, pp. 1207-1223, 2005.

[34] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy," *Proc. 14th Int'l Joint Conf. AI*, 1995.

[35] M.A. Rodriguez and M.J. Egenhofer, "Determining Semantic Similarity among Entity Classes from Different Ontologies," *IEEE Trans. Knowledge and Data Eng.*, vol. 15, no. 2, pp. 442-456, Mar./Apr. 2003.

[36] H. Rubenstein and J.B. Goodenough, "Contextual Correlates of Synonymy," *Comm. ACM*, vol. 8, no. 10, pp. 627-633, 1965.

[37] G. Salton and M. Lesk, *Computer evaluation of indexing and text processing*. Prentice Hall, Ing. Englewood Cliffs, New Jersey, pp. 143–180., 1971.

[38] P. Turney, "Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL," *Proc. Twelfth European Conf. Machine Learning*, 2001.

[39] Z. Wu and M. Palmer, "Verb Semantics and Lexical Selection," *Proc. Ann. Meeting Association for Computational Linguistics*, 1994.