Semantic Text Similarity Using Corpus-Based Word Similarity and String Similarity

AMINUL ISLAM and DIANA INKPEN
University of Ottawa

We present a method for measuring the semantic similarity of texts using a corpus-based measure of semantic word similarity and a normalized and modified version 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. We focus on computing the similarity between two sentences or 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.

Categories and Subject Descriptors: I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Natural language interfaces*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Semantic similarity of words, similarity of short texts, corpus-based measures

ACM Reference Format:

Islam, A. and Inkpen, D. 2008. Semantic text similarity using corpus-based word similarity and string similarity. ACM Trans. Knowl. Discov. Data. 2, 2, Article 10 (July 2008), 25 pages. DOI = 10.1145/1376815.1376819 http://doi.acm.org/10.1145/1376815.1376819

1. INTRODUCTION

Similarity is a complex concept which has been widely discussed in the linguistic, philosophical, and information theory communities [Hatzivassiloglou et al. 1999]. Frawley [1992] discusses all semantic typing in terms of two mechanisms: the detection of similarities and differences. Jackendoff [1983] argues that standard semantic relations such as synonymy, paraphrase, redundancy, and entailment all result from judgments of likeness whereas antonymy,

This research was supported by the Natural Sciences and Engineering Research Council of Canada.
Authors' address: School of Information Technology and Engineering, The University of Ottawa, 800 King Edward, Ottawa, ON, K1N 6N5, Canada; email: {mdislam, diana}@site.uottawa.ca.

contradiction, and inconsistency derive from judgments of difference. 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 to be one of the best techniques for improving retrieval effectiveness [Park et al. 2005] 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 [Coelho et al. 2004]. The use of text similarity is beneficial for relevance feedback and text categorization [Ko et al. 2004; Liu and Guo 2005], text summarization [Erkan and Radev 2004; Lin and Hovy 2003], word sense disambiguation [Lesk 1986; Schutze 1998], methods for automatic evaluation of machine translation [Liu and Zong 2004; Papineni et al. 2002], evaluation of text coherence [Katarzyna and Szczepaniak 2005; Lapata and Barzilay 2005], formatted documents classification [Katarzyna and Szczepaniak 2005].

In databases, text similarity can be used in schema matching to solve semantic heterogeneity, a key problem in any data sharing system whether it is a federated database, a data integration system, a message passing system, a web service, or a peer-to-peer data management system [Madhavan et al. 2005]. It can also be used in text similarity join operator that joins two relations if their join attributes are textually similar to each other, and it has a variety of application domains including integration and querying of data from heterogeneous resources; cleansing of data; and mining of data [Cohen 2000; Schallehn et al. 2004].

In the next section, we point out some drawbacks of the existing methods. One of the major drawbacks of most of the existing methods is the domain dependency, that is, once the similarity method is designed for a specific application domain, it cannot be adapted easily to other domains. This lack of adaptability to the domain does not correspond to human language usage as sentence meaning may change, to varying extents, from domain to domain. To address this drawback, we aim to develop a method that is fully automatic without requiring users' feedback and can be used independently of the domain in applications requiring small text or sentence similarity measure. The computation of text similarity can be viewed as a generic component for the research community dealing with text-related knowledge representation and discovery.

This article is organized as follows: Section 2 presents a brief overview of the related work. Our proposed method is described in Section 3. A walk-through example of the method is presented in Section 4. Evaluation and experimental results are discussed in Section 5. We address some contributions and future related work in Section 6.

2. RELATED WORK

There is extensive literature on measuring the similarity between long texts or documents [Hatzivassiloglou et al. 1999; Landauer and Dumais 1997;

Maguitman et al. 2005; Meadow et al. 2000], but there is less work related to the measurement of similarity between sentences or short texts [Foltz et al. 1998]. 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 [Meadow et al. 2000], 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 [Salton and Lesk 1971]. One extension of word co-occurrence methods leads to the pattern matching methods which are commonly used in text mining and conversational agents [Corley and Mihalcea 2005]. This technique relies on the assumption that more similar documents have more words in common. But it is not always the case that texts with similar meaning necessarily share many words. Again, the sentence representation is not very efficient as the vector dimension is very large compared to the number of words in a short text or sentence, thus, the resulting vectors would have many null components.

The Latent Semantic Analysis (LSA) [Landauer and Dumais 1997; Landauer et al. 1998] and the Hyperspace Analogues to Language (HAL) model [Burgess et al. 1998] are two well-known methods in corpus-based similarity. LSA, a high-dimensional linear association model, analyzes a large corpus of natural language text and generates a representation that captures the similarity of words and text passages. The underlying idea is that the aggregation of all the word contexts in which a given word does or does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other [Landauer et al. 1998]. The model tries to answer how people acquire as much knowledge as they do on the basis of as little information as they get. It uses Singular Value Decomposition (SVD) to find the semantic representations of words by analyzing the statistical relationships among words in a large corpus of text. When LSA is used to compute sentence similarity, a vector for each sentence is formed in the reduced-dimensional space; similarity is then measured by the cosine of the angle between their corresponding row vectors [Foltz et al. 1998]. The dimension size of the word by context matrix is limited and fixed to several hundred because of the computational limit of SVD. As a result the vector is fixed and is thus likely to be a very sparse representation of a short text such as a sentence. LSA does not take into account any syntactic information and is thus more appropriate for longer texts.

Another corpus-based method is Hyperspace Analogues to Language (HAL) [Burgess et al. 1998]. The HAL method uses lexical co-occurrence to produce a high-dimensional semantic space. A semantic space is a space in which words are represented as points, and the position of each word along the axes is related to the word's meaning. Once the space is constructed, a distance measure can be used to determine relationships between words. In HAL, this space is constructed by first passing a window over a large corpus and recording weighted lexical co-occurrences (words closer to the target word are given a higher weight than words farther away). These results are recorded in an $n \times n$ co-occurrence matrix with one row and one column for each unique word appearing in the corpus. Once this is complete, a vector representing each word in $2n$ dimensional space is formed by concatenating the transpose of a word's column to its row. Subsequently, a sentence vector is formed by adding together the word vectors for all words in the sentence. Similarity between two sentences is calculated using a metric such as Euclidean distance. However, the authors' experimental results showed that HAL was not as promising as LSA in the computation of similarity for short texts [Burgess et al. 1998]. HAL's drawback may be due to the building of the memory matrix and its approach to forming sentence vectors: The word-by-word matrix does not capture sentence meaning well and the sentence vector becomes diluted as a large number of words are added to it [Li et al. 2006].

Hybrid methods use both corpus-based measures [Turney 2001] and knowledge-based measures [Leacock and Chodorow 1998; Wu and Palmer 1994] of word semantic similarity to determine the text similarity. Mihalcea et al. [2006] 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) [Turney 2001] (details in Section 3.2) and LSA (Latent Semantic Analysis) [Landauer et al. 1998] and six knowledge-based measures [Jiang and Conrath 1997; Leacock and Chodorow 1998; Lesk 1986; Lin 1998; Resnik 1995; Wu and Palmer 1994] 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. [2006] 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 [Miller et al. 1993]. 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. These two hybrid measures [Li et al. 2006; Mihalcea et al. 2006] do not take into account the string similarity, which plays an important role in some cases. We discuss why string similarity is important in next section.

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 determines the similarity of two texts from semantic and syntactic information (in terms of common-word order) that they contain. We consider three similarity functions in order to derive a more generalized text similarity method. First, string similarity and semantic word similarity are calculated and then we use an optional common-word order similarity function to incorporate syntactic information in our method, if we wish. Finally, the text similarity is derived by combining string similarity, semantic similarity and common-word order 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. Even if we obtain any similarity value using corpus-based similarity measures, we obtain a very low similarity score. 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) [Allison and Dix 1986] 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] Kondrak [2005] showed that edit distance and the length of the longest common subsequence are special cases of n-gram distance and similarity, respectively. Melamed [1999] 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 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 the length of both the shorter and the longer string and call it

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

normalized longest common subsequence (NLCS) which is,

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

While in classical LCS, the common subsequence needs not be consecutive, in database schema matching, consecutive common subsequence is important for a high degree of matching. We use *maximal consecutive longest common subsequence* starting at character 1, $MCLCS_1$ (Algorithm 1) and maximal consecutive longest common subsequence starting at any character n, $MCLCS_n$ (Algorithm 2). In Algorithm 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 Algorithm 2, we present another 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 may start from any character (character n) for both of the strings. We normalize $MCLCS_1$ and $MCLCS_n$ and call it *normalized* $MCLCS_1$ ($NMCLCS_1$) and *normalized* $MCLCS_n$ ($NMCLCS_n$), respectively.

$$v_2 = NMCLCS_1(r_i, s_j) = \frac{length(MCLCS_1(r_i, s_j))^2}{length(r_i) \times length(s_j)} \tag{2}$$

$$v_3 = NMCLCS_n(r_i, s_j) = \frac{length(MCLCS_n(r_i, s_j))^2}{length(r_i) \times length(s_j)}. \tag{3}$$

We take the weighted sum of these individual 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:

$$\alpha = w_1v_1 + w_2v_2 + w_3v_3 \tag{4}$$

Algorithm 1. $MCLCS_1$ (Maximal Consecutive LCS starting at character 1)

```
 input : $r_i, s_j$        /*$r_i$ and $s_j$ are two input strings where $|r_i| = \tau$,
         $|s_j| = \eta$ and $\tau \leq \eta$  */
 output: $r_i$             /*$r_i$ is the Maximal Consecutive LCS starting at
         character 1 */
1 $\tau \leftarrow |r_i|,\ \eta \leftarrow |s_j|$
2 while $|r_i| \geq 0$ do
3    If $r_i \cap s_j$ then                    /* i.e., $r_i \subset s_j = r_i$*/
4       return $r_i$
5    else
6       $r_i \leftarrow r_i \backslash c_\tau$ /* i.e., remove the right most character from $r_i$ */
7    end
8 end
```

Algorithm 2. *$MCLCS_n$* (Maximal consecutive LCS starting at any character n)

```
input : $r_i$, $s_j$   /* $r_i$ and $s_j$ are two input strings where $|r_i| = \tau$,
          $|s_j| = \eta$ and $\tau \leq \eta$ */
output: $x$   /*$x$ is the Maximal Consecutive LCS starting at any
          character $n$ */
1  $\tau \leftarrow |r_i|$, $\eta \leftarrow |s_j|$
2  while $|r_i| \geq 0$ do
3      determine all $n$-grams from $r_i$ where $n = 1 \ldots |r_i|$ and
4      $\overline{r_i}$ is the set of $n$-grams
5      If $x \in s_j$ where $\{x | x \in \overline{r_i},\ x = Max(\overline{r_i})\}$ then   /* $i$ is the number of $n$-grams
       and $Max(\overline{r_i})$ returns the maximum length $n$-gram from $\overline{r_i}$
       */
6          return $x$
7      else
8          $\overline{r_i} \leftarrow \overline{r_i} \backslash x$
                                                  /*remove $x$ from $\overline{r_i}$ */
9      end
10 end
```

We set equal weights for our experiments.[2] Theoretically, $v_3 \geq v_2$. For example, if $r_i = albastru$ and $s_j = alabaster$, then

$$LCS(r_i, s_j) = albastr$$
$$MCLCS_1(r_i, s_j) = al$$
$$MCLCS_n(r_i, s_j) = bast$$
$$NLCS(r_i, s_j) = 7^2/(8 \times 9) = 0.68$$
$$NMCLCS_1 = 2^2/(8 \times 9) = 0.056$$
$$NMCLCS_n(r_i, s_j) = 4^2/(8 \times 9) = 0.22$$

$$\text{The string similarity, } \alpha = w_1 v_1 + w_2 v_2 + w_3 v_3 = 0.33 \times 0.68 + 0.33 \times 0.056 + 0.33 \times 0.22 = 0.32$$

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-based (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 Li et al. [2003], Rodriguez and Egenhofer [2003], Weeds et al. [2004], and Bollegala et al. [2007]. We focus our attention on corpus-based measures because of their large type coverage. The types that are used in real-world texts are often not found in knowledge base.

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

PMI-IR [Turney 2001] is a simple method for computing corpus-based similarity of words. It uses Pointwise Mutual Information, defined as follows:

$\text{PMI}(w_1, w_2) = \log p(w_1 \text{ AND } w_2)/p(w_1)p(w_2)$ Here, w_1 and w_2 are the two words. $p(w_1 \text{ AND } w_2)$ is the probability that the two words co-occur. If w_1 and w_2 are statistically independent, then the probability that they co-occur is given by the product $p(w_1) \cdot p(w_2)$. If they are not independent, and they have a tendency to co-occur, then $p(w_1 \text{ AND } w_2)$ will be greater than $p(w_1) \cdot p(w_2)$.

PMI-IR used AltaVista Advanced Search query syntax to calculate the probabilities.[3] In the simplest case, two words co-occur when they appear in the same document. The probabilities can be approximated by the number of documents retrieved:

$\text{PMI-IR}(w_1, w_2) = \mathit{hits}(w_1 \text{ AND } w_2)/(\mathit{hits}(w_1)\mathit{hits}(w_2))$
Here, $\mathit{hits}(x)$ be the number of hits (the number of documents retrieved) when the query x is given to AltaVista. AltaVista provides how many documents contain both w_1 and w_2, and then how many documents contain w_1 alone, and how many documents contain w_2 alone.

LSA, another corpus-based measure, analyzes a large corpus of natural text and generate a representation that captures the similarity of words (discussed in Section 2).

We use the Second Order Co-occurrence PMI (SOC-PMI) word similarity method [Islam and Inkpen 2006] that uses Pointwise Mutual Information to sort lists of important neighbor words of the two target words from a large corpus. PMI-IR used AltaVista's Advanced Search query syntax to calculate probabilities. Note that the "NEAR" search operator of AltaVista is an essential operator in the PMI-IR method. However, it is no longer in use in AltaVista; this means that, from the implementation point of view, it is not possible to use the PMI-IR method in the same form in new systems [Islam et al. 2008]. In any case, from the algorithmic point of view, the advantage of using SOC-PMI in our system is that it can calculate the similarity between two words that do not co-occur frequently, because they co-occur with the same neighboring words. We used the British National Corpus (BNC)[4] as a source of frequencies and contexts. The method considers the words that 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 $f^b(t_i, w) > 0$,

$$f^{pmi}(t_i, w) = \log_2 \frac{f^b(t_i, w) \times m}{f^t(t_i) f^t(w)}, \tag{5}$$

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[5] and m is total number of tokens in the corpus. Now, for word w, we

[3]Google and other well-known search engines do not have a search feature similar to AltaVista's "NEAR" operator.

[4]The size of this corpus is approximately 100 million words, and it is a balanced corpus: it contains texts from various sources, general British English. For details, see http://www.natcorp.ox.ac.uk/.

[5]The size of the window is 11 words. For example, a window of size 11 will have at most five context words on either side of the middle word (which is actually the word of interest).

define a set of words, X^w, sorted in descending order by their PMI values with w and taken the top most β words having $f^{pmi}(t_i, w) > 0$.

$$X^w = \{X_i^w\}, \text{ where } i = 1, 2, \ldots, \beta \text{ and}$$
$$f^{pmi}(t_1, w) \geq f^{pmi}(t_2, w) \geq \cdots \geq f^{pmi}(t_{\beta-1}, w) \geq f^{pmi}(t_\beta, w)$$

A rule of thumb is used to choose the value of β.[6] We define the *β-PMI summation* function of a word with respect to another word. The *β-PMI summation* function for word w_1 with respect to word w_2 is:

$$f(w_1, w_2, \beta) = \sum_{i=1}^{\beta} \left(f^{pmi}\left(X_i^{w_1}, w_2\right)\right)^{\gamma}, \tag{6}$$

where, $f^{pmi}(X_i^{w_1}, w_2) > 0$, which sums all the positive PMI values of words in the set X^{w_2} also common to the words in the set X^{w_1}. 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. γ[7] should have a value greater than 1. So, the *β-PMI summation* function for word w_1 with respect to word w_2 having $\beta = \beta_1$ and the *β-PMI summation* function for word w_2 with respect to word w_1 having $\beta = \beta_2$ are $f(w_1, w_2, \beta_1) = \sum_{i=1}^{\beta_1}(f^{pmi}(X_i^{w_1}, w_2))^{\gamma}$ and $f(w_2, w_1, \beta_2) = \sum_{i=1}^{\beta_2}(f^{pmi}(X_i^{w_2}, w_1))^{\gamma}$, respectively.[8] Finally, we define the *semantic PMI similarity* function between the two words, w_1 and w_2,

$$Sim(w_1, w_2) = \frac{f(w_1, w_2, \beta_1)}{\beta_1} + \frac{f(w_2, w_1, \beta_2)}{\beta_2}. \tag{7}$$

We normalize the semantic word similarity, so that it provides a similarity score between 0 and 1 inclusively. The normalization of semantic similarity algorithm (Algorithm 3) returns a normalized score of similarity between two words. It takes as arguments the two words, r_i and s_j, and a maximum value, λ, that is returned by the semantic similarity function, $Sim()$. It returns a similarity score between 0 and 1 inclusively. For example, the algorithm returns 0.986 for words *cemetery* and *graveyard* with $\lambda = 20$ (for SOC-PMI method). 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, if someone wants to try other word-similarity methods (dictionary-based, corpus-based or hybrid). In that case, we need to set λ to the maximum similarity value specific to that method. For example, if we use the Roget-based

[6]The value of β is related to how many times the word, w appears in the corpus (i.e., the frequency of w) as well as the number of types (n) in the corpus. We define β as $\beta = (\log(f^t(w)))^2 \frac{\log_2(n)}{\mu}$, where μ is a constant. For all of our experiments we used $\mu = 6.5$. The value of μ depends on the size of the corpus. The smaller the corpus we use, the smaller the value of μ we should choose. If we lower the value of μ, we lose some important/interesting words, and if we increase it we consider more words and this significantly degrades the result.

[7]The higher the value of γ is, the greater emphasis on words having very high PMI values with w is given. For all our experiments, we chose $\gamma = 3$. The value $\gamma \geq 4$ is not a good choice because it puts too much emphasis on words that have very high PMI values with w and ignores all the words having moderate or low PMI values. We experimented on a small portion of the BNC to find out the values of γ, β and μ.

[8]β_1 and β_2 are the number of neighbours for w_1 and w_2 that we take into account.

Algorithm 3. Normalization of Semantic similarity matching

```
  input : $r_i$ , $s_j$ , $\lambda$      /* $r_i$ and $s_j$ are two input words where $|r_i| = \tau$ , $|s_j| = \eta$
          and $\tau \leq \eta$
  output: $v$ /* $v$ is the semantic similarity value between 0 and 1,
          inclusively */
1 $v \leftarrow Sim(r_i, s_j)$   /* This function determines semantic similarity between
  two words using (8). Details of this method have been discussed in
  Section 3.2. Any other similarity method can also be used instead.  /*
2 If $v > \lambda$ then   /* $\lambda$ is the maximum possible similarity value as discussed
  in section 3.2 */
3     $v \leftarrow 1$
4 else
5     $v \leftarrow v/\lambda$
6 end
```

measure [Jarmasz and Szpakowicz 2003], then we need to set λ to 16, as there are eight levels in the Roget thesaurus and there can be at most 16 edges between two words. One of the main advantages of using distributional measures based on corpus is that it covers significantly more tokens than any dictionary-based measure.

3.3 Common Word Order Similarity between Sentences

If the two texts have some words in common, we can measure how similar the order of the common-words is in the two texts (if these words appear in the same order, or almost the same order, or very different order). Although syntactic information (here approximated by the order of the common-words) has low importance for the semantic processing of short texts according to Wiemer-Hastings [2000], we incorporate the word order to test this hypothesis and to make our method more generic. We use it when we consider the importance of syntactic information by setting its weight factor, w_f to less than 0.5, that is, $w_f \in [0, 0.5)$. We set its weight factor, w_f to 0, when we want to ignore its importance. The value of w_f should be much less than 0.5, as syntax has a little importance in semantic processing.

Let us consider a pair of sentences, P and R has m and n tokens respectively, that is, $P = p_1, p_2, \ldots, p_m$ and $R = r_1, r_2, \ldots, r_n$ and $n \geq m$. Otherwise, we switch P and R. We count the number of p_i's (say, δ) for which $p_i = r_j$, for all $p \in P$ and for all $r \in R$. That is, there are δ tokens in P that exactly match with R, where $\delta \leq m$. We remove all δ tokens from P and put them in X and from R in Y, in the same order as they appear in the sentences. So, $X = \{x_1, x_2, \ldots, x_\delta\}$ and $Y = \{y_1, y_2, \ldots, y_\delta\}$. We replace X by assigning a unique index number for each token in X starting from 1 to δ, that is, $X = \{1, 2, \ldots, \delta\}$. Based on this unique index numbers for each token in X, we also replace Y where $X = Y$. We propose a measure for measuring the common-word order similarity of two texts as:

$$S_o = 1 - \frac{|x_1 - y_1| + |x_2 - y_2| + \cdots + |x_\delta - y_\delta|}{|x_1 - x_\delta| + |x_2 - x_{\delta-1}| + \cdots + |x_\delta - x_1|} \tag{8}$$

That is, common-word order similarity is determined by the normalized difference of common-word order (the denominator is the largest possible dissimilarity value, the worse order of pairwise elements in X). Equation (9) demonstrates three individual cases of (8).[9]

$$S_o = \begin{cases} 1 - \frac{2\sum_{i=1}^{\delta}|x_i - y_i|}{\delta^2} & \text{if } \delta \text{ is even} \\ 1 - \frac{2\sum_{i=1}^{\delta}|x_i - y_i|}{\delta^2 - 1} & \text{if } \delta \text{ is odd and } \delta > 1 \\ 1 & \text{if } \delta \text{ is odd and } \delta = 1. \end{cases} \tag{9}$$

For example:
P : *Many consider Maradona as the best player in soccer history.*
R : *Maradona is one of the best soccer players.*

There are five tokens (δ) in P that exactly match with R. We remove all five tokens from both P and R to X and Y in the same order as they appear in the sentences. So, $X = \{$*maradona, the, best, player, soccer*$\}$ and $Y = \{$*maradona, the, best, soccer, player*$\}$. We replace X by assigning a unique index number for each token in X starting from 1 to 5, that is, $X = \{1, 2, 3, 4, 5\}$. Based on this unique index numbers for each token in X, we also replace Y where $X = Y$. That is, $Y = \{1, 2, 3, 5, 4\}$.

$$S_o = 1 - \frac{2\sum_{i=1}^{5}|x_i - y_i|}{5^2 - 1}, \text{that is, } S_o = 0.83 \text{ as } \delta \text{ is odd and } \delta > 1.$$

3.4 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 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 $r \in R$. That is, 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, that is, $p_i = \{c_1c_2 \cdots c_\tau\}$ and any token $r_j \in R$ has η characters, that is, $r_j = \{c_1c_2 \cdots c_\eta\}$ where $\tau \leq \eta$. In other words, η is the length of the longer

[9]The denominator is different for δ even or odd. For example, if δ is even and $X = \{1, 2, 3, 4, 5, 6\}$ then the denominator is $|1-6|+|2-5|+|3-4|+|4-3|+|5-2|+|6-1| = 5+3+1+1+3+5 = \delta^2/2$. If δ is odd and $X = \{1, 2, 3, 4, 5\}$, then the denominator is $|1-5|+|2-4|+|3-3|+|4-2|+|5-1| = 4+2+0+2+4 = (\delta^2 - 1)/2$.

token and τ is the length of the shorter token. We calculate the following:

$v_1 \leftarrow NLCS(p_i, r_j)$
$v_2 \leftarrow NMCLCS_1(p_i, r_j)$
$v_3 \leftarrow NMCLCS_n(p_i, r_j)$
$\alpha_{ij} \leftarrow w_1 v_1 + w_2 v_2 + w_3 v_3,$

that is, 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 $i = 1 \cdots m - \delta$ and $j = 1 \cdots n - \delta$.

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

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} ($\beta_{ij} \leftarrow semanticMatching(p_i, r_j)$ (Algorithm 3) in row i and column j position of the matrix for all $i = 1 \cdots m - \delta$ and $j = 1 \cdots n - \delta$.

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

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

$$M \leftarrow \psi M_1 + \varphi M_2, \tag{10}$$

(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 $\psi + \varphi = 1$. Setting any one of these factors to 0 means that we do not include that matrix. Setting both of the factors to 0.5 means we consider them equally important.

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

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 $\gamma_{ij} \geq 0$. We remove all the matrix elements of ith row and jth column from M.[10] 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 ρ and add $\delta(1 - w_f + w_f S_o)$ to it to get a total score, where S_o is common-word order similarity score and w_f is common-word order weight that decides the relative contributions of word similarity (string similarity and semantic similarity) and common-word order similarity 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.

$$S(P,R) = \frac{\left(\delta(1 - w_f + w_f S_o) + \sum_{i=1}^{|\rho|} \rho_i\right) \times (m+n)}{2mn} \tag{11}$$

(11) could take four specific forms: First, if we ignore the importance of syntactic information by setting $w_f = 0$ in (11), we obtain:

$$S(P,R) = \frac{\left(\delta + \sum_{i=1}^{|\rho|} \rho_i\right) \times (m+n)}{2mn} \tag{12}$$

Second, if we obtain common-word order similarity value (S_o) as 1, $S(P,R)$ will be independent of w_f, that is, same as (12). Third, if we ignore the importance of string similarity, we set ψ in (10) to 0. Fourth, if we ignore the importance of semantic word similarity, we set φ in (10) to 0.

4. A WALK THROUGH EXAMPLE

Let P = "*A cemetery is a place where dead people's bodies or their ashes are buried.*", R = "*A graveyard is an area of land, sometimes near a church, where dead people are buried.*". This example is from the dataset used by Li et al. [2006] (see the description of the full dataset in Section 5.1).

Step 1. After eliminating all special characters and punctuations, if any, and then removing all stop words and lemmatizing, we get P = {*cemetery*, *place*, *where*, *dead*, *body*, *ash*, *bury*} and R = {*graveyard*, *area*, *land*, *sometime*, *near*, *church*, *where*, *dead*, *bury*} where $m = 7$ and $n = 9$.

Step 2. Only three tokens (i.e., *where*, *dead* and *bury*) in P exactly matches with R therefore we set δ to 3. We remove *where*, *dead* and *bury* from both P and R. So, P = {*cemetery*, *place*, *body*, *ash*} and R = {*graveyard*, *area*, *land*, *sometime*, *near*, *church*}. As $m - \delta \neq 0$, we proceed to next step.

Step 3. We construct a 4×6 *string matching matrix*, M_1. Consider the *place land* pair where *length*(*LCS*(*place*, *land*)) = 2, $\eta = 5$ is the length of the longer token (*place*), $\tau = 4$ is the length of the shorter token (*land*) and 2 is the maximal length of the consecutive portions of the shorter token that consecutively match

[10]We remove the row and column in order to remove the pair with maximum similarity. This makes the computation manageable: in the next steps, fewer words are left for matching.

with the longer token, where matching starts from the 1st character of the shorter token and 2nd character of the longer token. So,
$v_1 = 22/(5 \times 4) = 0.2, \quad v_2 = 0, \quad v_3 = 22/(5 \times 4) = 0.2$
and $\alpha_{23} = 0.33 \times v_1 + 0.33 \times v_2 + 0.33 \times v_3 = 0.132$

$$M_1 = \begin{array}{c} \\ cemetery \\ place \\ body \\ ash \end{array} \begin{array}{cccccc} graveyard & area & land & sometime & near & church \\ \end{array}$$

$$M_1 = \begin{array}{c|cccccc} & graveyard & area & land & sometime & near & church \\ \hline cemetery & 0.023 & 0.021 & 0 & 0.129 & 0.052 & 0.041 \\ place & 0.037 & 0.083 & 0.132 & 0.017 & 0.033 & 0.022 \\ body & 0.018 & 0 & 0.041 & 0.021 & 0 & 0 \\ ash & 0.024 & 0.083 & 0.055 & 0.028 & 0.055 & 0.037 \end{array}.$$

Step 4. We construct a 4×6 *semantic similarity matrix*, M_2. Here, $\lambda = 20$ as we used *SOCPMI* method.

$$M_2 = \begin{array}{c|cccccc} & graveyard & area & land & sometime & near & church \\ \hline cemetery & 0.986 & 0 & 0.390 & 0.195 & 0.542 & 0.856 \\ place & 0 & 0.413 & 0.276 & 0.149 & 0 & 0 \\ body & 0.465 & 0 & 0.363 & 0.122 & 0.063 & 0.088 \\ ash & 0.796 & 0 & 0.213 & 0.238 & 0.395 & 0.211 \end{array}.$$

Step 5. We construct a 4×6 *joint matrix*, M and assign equal weight factor by setting both ψ and φ to 0.5.

$$M = \begin{array}{c|cccccc} & graveyard & area & land & sometime & near & church \\ \hline cemetery & \mathbf{0.505} & 0.010 & 0.195 & 0.162 & 0.297 & 0.449 \\ place & 0.018 & 0.248 & 0.204 & 0.083 & 0.017 & 0.011 \\ body & 0.242 & 0 & 0.039 & 0.071 & 0.032 & 0.044 \\ ash & 0.416 & 0.041 & 0.134 & 0.133 & 0.225 & 0.124 \end{array}.$$

We find the maximum-valued matrix-element, $\gamma_{ij} = 0.505$ and add it to ρ as $\gamma_{ij} \geq 0$. So, $\rho = \{0.505\}$. The new M after removing ith (i = 1) row and jth (j = 1) column is:

$$M = \begin{array}{c|ccccc} & area & land & sometime & near & church \\ \hline place & \mathbf{0.248} & 0.204 & 0.083 & 0.017 & 0.011 \\ body & 0 & 0.039 & 0.071 & 0.032 & 0.044 \\ ash & 0.041 & 0.134 & 0.133 & 0.225 & 0.124 \end{array}.$$

We find the maximum-valued matrix-element, $\gamma_{ij} = 0.248$ for this new M and add it to ρ as $\gamma_{ij} \geq 0$. So, $\rho = \{0.505, 0.248\}$. The new M after removing ith ($i = 1$) row and jth ($j = 1$) column is:

$$M = \begin{array}{c|cccc} & land & sometime & near & church \\ \hline body & 0.039 & 0.071 & 0.032 & 0.044 \\ ash & 0.134 & 0.133 & \mathbf{0.225} & 0.124 \end{array}.$$

Here, 0.225 is the maximum-valued matrix-element and $\gamma_{ij} \geq 0$. So, $\rho = \{0.505, 0.248, 0.225\}$. The new M after removing ith ($i = 2$) row and jth ($j = 3$) column is:

$$M = \begin{array}{c|ccc} & land & sometime & church \\ \hline body & 0.039 & \mathbf{0.071} & 0.044 \end{array}.$$

We find 0.071 as the maximum-valued matrix-element and $\gamma_{ij} \geq 0$. So, $\rho = \{0.505, 0.248, 0.225, 0.071\}$. The new M is empty after removing ith ($i = 1$) row and jth ($j = 2$) column.

We proceed to next step as $m - \delta - |\rho| = 0$. (Here, $m = 7$, $\delta = 3$ and $|\rho| = 4$)
Step 6.

$$\begin{aligned} S(P, R) &= \frac{\left(\delta + \sum_{i=1}^{|\rho|} \rho_i\right) \times (m + n)}{2mn} \\ &= (3 + 1.049) \times 16/126 = 0.514. \end{aligned}$$

For this specific example, it does not matter what value we choose for w_f (common-word order factor), because we obtain common-word order similarity value as 1. That is, $S(P, R) = 0.514$ for all $w_f \in [0, 0.5)$.

5. EVALUATION AND EXPERIMENTAL RESULTS

In order to evaluate our text similarity measure, we use two different data sets. In our first experiment, we compute the similarity score for 30 sentence pairs and find the correlation with human judges in order to compare with Li et al. [2006] who also use the same 30 sentence pairs and find the correlation with human judges. In our second experiment, we use the Microsoft paraphrase corpus [Dolan et al. 2004], consisting of 4,076 training and 1,725 test pairs, and determine the number of correctly identified paraphrase pairs in the corpus using our proposed semantic text similarity method and compare the result with Mihalcea et al. [2006] and Corley and Mihalcea [2005] as they also use the same data set to evaluate their method.

Since syntax plays a less important role in semantic processing of shorter texts, the results of the following experiments are for the common-word order similarity factor set to zero (if set to a higher value the results are slightly lower).

5.1 Experiment with Human Similarities of Sentence Pairs

We use the same data set as Li et al. [2006].[11] Li et al. [2006] 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. [2006] began with the set of 65 noun pairs from Rubenstein and Goodenough [1965] and replaced them with their definitions from the Collins Cobuild dictionary [Sinclair 2001]. 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 [1965]. 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.

[11]Available at http://www.docm.mmu.ac.uk/STAFF/D.McLean/SentenceResults.htm.

Table I. Sentence Data Set Results

R&G No.	R&G Word Pair in the Sentences	Human Similarity (Mean)	Li et al. Similarity Similarity Method	Semantic Text Similarity Method
1	Cord Smile	0.01	0.33	0.06
5	Autograph Shore	0.01	0.29	0.11
9	Asylum Fruit	0.01	0.21	0.07
13	Boy Rooster	0.11	0.53	0.16
17	Coast Forest	0.13	0.36	0.26
21	Boy Sage	0.04	0.51	0.16
25	Forest Graveyard	0.07	0.55	0.33
29	Bird Woodland	0.01	0.33	0.12
33	Hill Woodland	0.15	0.59	0.29
37	Magician Oracle	0.13	0.44	0.20
41	Oracle Sage	0.28	0.43	0.09
47	Furnace Stove	0.35	0.72	0.30
48	Magician Wizard	0.36	0.65	0.34
49	Hill Mound	0.29	0.74	0.15
50	Cord String	0.47	0.68	0.49
51	Glass Tumbler	0.14	0.65	0.28
52	Grin Smile	0.49	0.49	0.32
53	Serf Slave	0.48	0.39	0.44
54	Journey Voyage	0.36	0.52	0.41
55	Autograph Signature	0.41	0.55	0.19
56	Coast Shore	0.59	0.76	0.47
57	Forest Woodland	0.63	0.70	0.26
58	Implement Tool	0.59	0.75	0.51
59	Cock Rooster	0.86	1	0.94
60	Boy Lad	0.58	0.66	0.60
61	Cushion Pillow	0.52	0.66	0.29
62	Cemetery Graveyard	0.77	0.73	0.51
63	Automobile Car	0.56	0.64	0.52
64	Midday Noon	0.96	1	0.93
65	Gem Jewel	0.65	0.83	0.65

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 total of 46 out of 65 sentence pairs were rated 0.0 to 0.9 and the rest 19 pairs were rated 1.00 to 4.00). That is why a subset of 30 sentence pairs containing all of the 19 sentence pairs rated 1.0 to 4.0 and 11 (from the rest 46) sentence pairs rated 0.0 to 0.9 taken at equally spaced intervals from the list were selected to obtain a more even distribution across the similarity range based on a similar procedure to Miller and Charles [1991]. The detailed procedure of this data set preparation is in Li et al. [2006]. Table I shows human similarity scores along with Li et al.'s [2006], Similarity Method scores 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].

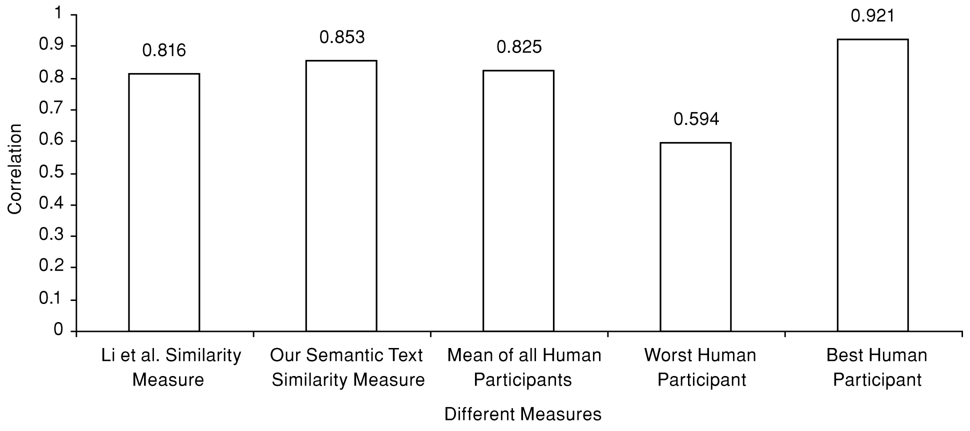

Fig. 1. Similarity correlations.

Figure 1 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 [2006], Similarity Measure achieves 0.816. The improvement we obtained is statistically significant at the 0.05 level.[12] The best participant obtained a correlation of 0.921 and the worst 0.594 with the average of the human judges that is used as expected solution to the task. Li et al. [2006] calculated the correlation coefficient for the judgments of each participant against the rest of the group and then took the mean to determine the mean of all participants which is 0.825 and considered this as the upper bound. We do not think this is necessarily an upper bound. The upper bound should be the correlation for the best human participant mentioned by Li et al. [2006], which is 0.921.

5.2 Experiment with the 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 [Dolan et al. 2004], consisting of 4,076 training and 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 automatically collected from thousands of news sources on the Web over a period of 18 months, and were subsequently 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. Table II summarizes the characteristics of the dataset and presents our experimental results.

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

Table II. 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 ($TP + FP$)
4076 (Training)	2753	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
		0.5	69.53	2708	3905
		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
1725 (Test)	1147	0	66.49	1147	1725
		0.1	66.49	1147	1725
		0.2	66.49	1147	1725
		0.3	66.49	1147	1725
		0.4	66.66	1146	1720
		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

Here are two examples from the Microsoft paraphrase corpus. The first example show two sentences that are labeled by the human judges as paraphrases and the second example shows two sentences that are not paraphrases:

Example 1:
T_1: Now, with the agency's last three shuttles grounded in the wake of the Columbia disaster, that wait could be even longer.
T_2: With the remaining three shuttles grounded in the wake of the Columbia accident, the rookies will have to wait even longer.

Example 2:
T_1: Ballmer has been vocal in the past warning that Linux is a threat to Microsoft.
T_2: In the memo, Ballmer reiterated the open-source threat to Microsoft.

We acknowledge, as in Corley and Mihalcea [2005], 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 (e.g., 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, recall and F-measure. Recall is defined as the percentage of pairs in the manually annotated pairs set identified by the method and precision is defined as the

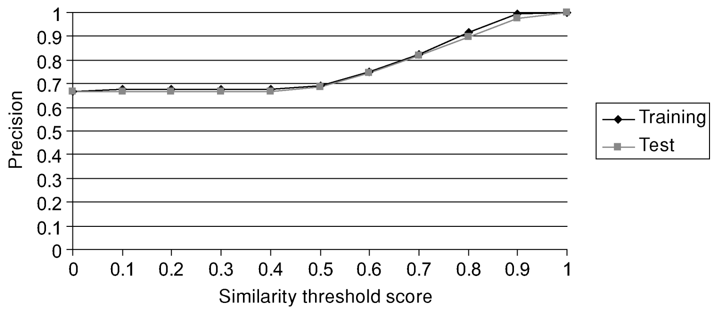

Fig. 2. Precision vs. similarity threshold curves of two data sets for eleven different similarity thresholds.

percentage of pairs returned by the method that also occurred in the manually annotated pairs set. In general, it is easy to obtain high performance for one of the two measures but relatively difficult to obtain high performance for both. F-measure (F) is the geometric mean of precision (P) and recall (R) and expresses a trade-off between those two measures. These performance measures are defined as follows:

$$\begin{aligned} P &= TP/(TP + FP) \\ R &= TP/(TP + FN) \\ F &= (1 + \beta)PR/(\beta P + R) \\ &= 2\text{PR} / (\text{P} + \text{R}), \end{aligned}$$

with $\beta = 1$ such that precision and recall weighted equally. Here, *TP*, *FP* and *FN* stand for True Positive (how many pairs of sentences were classified as paraphrases and they were indeed labeled as paraphrases in the data set), False Positive (how many were classified as nonparaphrases while they truly are paraphrases), and False Negative (how many were labeled as paraphrases when they should not have been), respectively.

We use eleven different similarity thresholds ranging from 0 to 1 with interval 0.1. For example, using test data set when we use similarity threshold 0.6, our method predicts 1369 pairs as correct, out of which 1022 pairs are correct among the 1725 manually annotated pairs. Precision vs. similarity threshold curves and recall vs. similarity threshold curves of the two data sets (training and test) for the eleven different similarity thresholds are shown in Figure 2 and Figure 3, respectively. Accuracy of the two data sets for the eleven different similarity thresholds is shown in Figure 4, whereas Figure 5 shows F-measure vs. similarity threshold curves.

In Figure 3, when we use a similarity threshold score of 1 (i.e., matching word by word exactly, therefore no semantic similarity matching is needed), we obtain recall values of 0.0054 and 0.0044 for the training and the test data set, respectively. We can consider these scores as one of the baselines. Mihalcea et al. [2006] mentioned two other baselines: Vector-based and Random. See Table III for the results of these baselines and the results of several methods from Mihalcea et al. [2006] and Corley and Mihalcea [2005] (on the test set).

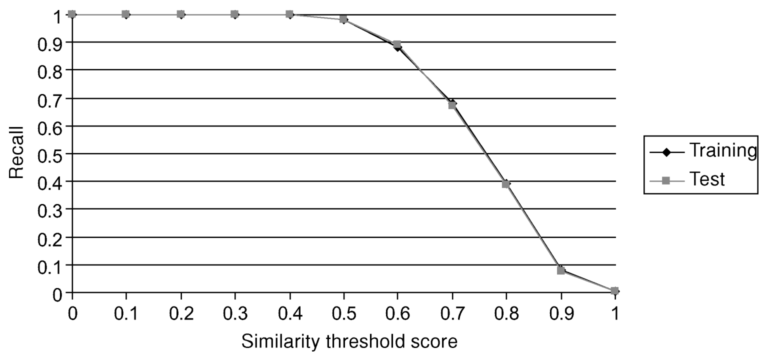

Fig. 3. Recall vs. similarity threshold curves of two data sets for eleven different similarity thresholds.

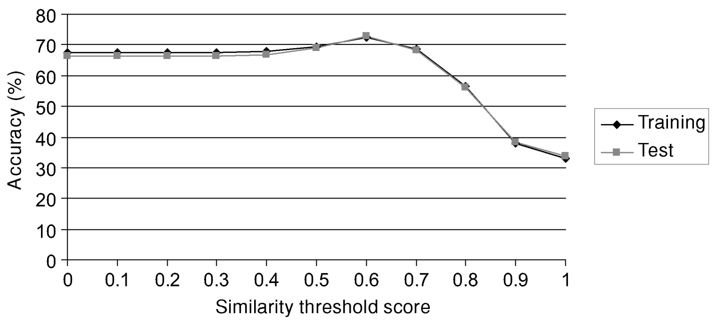

Fig. 4. Accuracy vs. similarity threshold curves of two data sets for eleven different similarity thresholds.

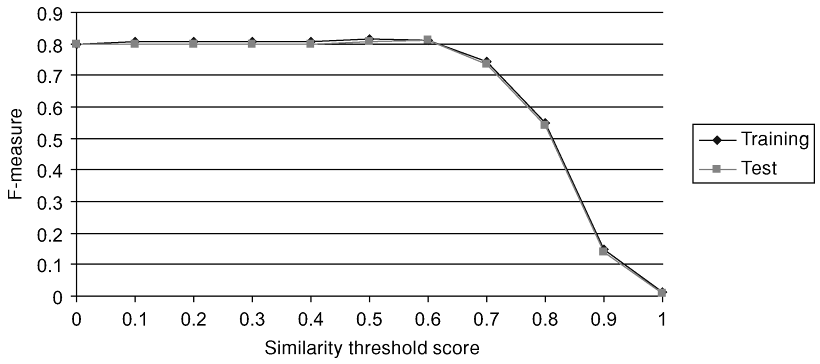

Fig. 5. F-measure vs. similarity threshold curves of two data sets for eleven different similarity threshold.

For this paraphrase identification task, we can consider our proposed STS method as a supervised method. Using the 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

Table III. Text Similarity for Paraphrase Identification

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

are correct among the 1725 manually annotated pairs). Our results on the test set are shown in Table III.

For each candidate paraphrase pair in the test set, we first calculate the semantic text similarity score using (12), and then label the candidate pair as a paraphrase if the similarity score exceeds a threshold of 0.6. The results on the test set are shown in Table III. We obtain the same F-measure (81%) at the combined methods from Mihalcea et al. [2006] and Corley and Mihalcea [2005]. We obtain higher accuracy and precision at the cost of decreasing recall.

6. CONCLUSION

6.1 Contributions of the Work

The proposed method determines the similarity of two texts from semantic and syntactic information (in terms of common-word order) that they contain. We consider two mandatory (string similarity and semantic word similarity) functions and an optional (common-word order similarity) function in order to derive a more generalized text similarity method. 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. [2006] (the improvement is statistically significant). For the paraphrase recognition task, our proposed STS method performs similarly to the combined unsupervised method [Mihalcea et al. 2006] and the combined supervised method [Corley and Mihalcea 2005]. The main advantage of our system is that it has lower time complexity than the other system [Mihalcea et al. 2006; Corley and Mihalcea 2005], because we use only one corpus-based measure, while they combine both corpus-based and WordNet-based measures. For example, Mihalcea et al. [2006] use six WordNet-based measures and two corpus-based measures. The time complexity of the algorithms is given mainly by the number of searches in the corpus and in

WordNet. One difference between our method and that of Mihalcea et al. [2006] with respect to time complexity is the size of the corpus. We use a corpus of size 10^8 whereas they use a web corpus of size 7×10^{11}. Another difference is that we do not use WordNet at all, therefore we save time complexity. The third difference is that we add the string similarity measure on short strings, but this is very fast, because the strings are short.[13]

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).

We also tested our proposed semantic text similarity method by setting common-word order similarity factor, $w_f \in [0, 0.5)$ to observe the effects of common-word order similarity on both data sets for determining short text similarity. For both the paraphrase identification and 30 sentence pair similarity tasks, we got lower accuracy and correlation with human ratings when using the word order similarity factor. This is similar to the conclusion of the word

[13]In our proposed method we do the following steps:

(i) First, we use three string similarity functions to determine a combined string similarity score. The time complexity calculation is straightforward. Assume that the maximum length of the two strings is m (where m is a small number, maximum 22 in the dataset). Then the time complexity of LCS, $MCLCS_1$ and $MCLCS_n$ are $O(m^2)$, $O(m^2)$ and $O(m^2)$, respectively. So, the total complexity of the string matching is $O(m^2)$.

(ii) We use one semantic word similarity method; it has a linear time complexity with the size of the corpus $N = 10^8$ (the size of the BNC). We can ignore quadratic time complexity of the window size (10 words) as it is much smaller than the size N of the corpus. So the total complexity of the corpus-based similarity matching is $O(N)$.

(iii) We use an optional common-word order similarity function to incorporate syntactic information in our method. It has a linear time complexity of the number of common words. We could omit this step as it lowers the accuracy and the correlation.

(iv) Finally, we combine the two matrices; this has a quadratic complexity of the size of the matrices. If s is the length of the longest sentence (s is 32 in the dataset), then the time complexity is $O(s^2)$.

The total complexity is $O(N) + O(m^2) + O(s^2)$.

Mihalcea et al. [2006] does the following steps:

(i) They use two corpus-based measures which have linear time complexity of the size of the corpus where the size of the corpus is $N' = 7 \times 10^{11}$ (a Web corpus). The complexity is $O(N')$.

(ii) They use six WordNet-based measures where the complexities of the WordNet-based measures are given mainly by the number of searches in WordNet. The theoretical complexity is proportional with the square of the number of senses of the two words, because we need to compare every sense of one word with every sense of the other word, and with the depth of WordNet hierarchy (h). If the maximum number of senses in WordNet is r, the time complexity of the WordNet-based similarity is $O(s^2 r^2 h)$, for the s^2 pairs of words. In practice, searches in WordNet are known to be time consuming.

(iii) Then they combine the corpus-based and WordNet-based measures.

(iv) Finally, they match the words in the two sentences. If s is the length of the longest sentence, the time complexity is $O(s^2)$, the same as in our method because we use the same data.

The total complexity is $O(N') + O(s^2 r^2 h) + O(s^2)$.

order experiments of Li et al. [2006] on the 30 sentence data set (note that they use a different method for dealing with word order, for all the words in the two sentences, not only the shared words). One of the possible reasons may be that the same meaning is expressed by different syntactic order in the short texts.

6.2 Future Work

A follow up study could use a topic directory such as Dmoz[14] (where about 590,000 categories are available) to collect category descriptions. Semantic similarity for pairs of category descriptions could be computed using our proposed method, because they are short texts. This approach would allow evaluation over a large number of pairs of topic descriptions, and we could estimate the expected similarity scores from the positions of the nodes in the topic ontologies available in Dmoz.

REFERENCES

Allison, L. and Dix, T. 1986. A bit-string longest-common-subsequence algorithm. *Inf. Proc. Lett. 23*, 305–310.

Bollegala, D., Matsuo, Y., and Ishizuka, M. 2007. Measuring semantic similarity between words using web search engines. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*. ACM, New York, 757–766.

Burgess, C., Livesay, K., and Lund, K. 1998. Explorations in context space: Words, sentences, discourse. *Disc. Proc. 25*, 2–3, 211–257.

Coelho, T., Calado, P., Souza, L., Ribeiro-Neto, B., and Muntz, R. 2004. Image retrieval using multiple evidence ranking. *IEEE Trans. Knowl. Data Eng. 16*, 4, 408–417.

Cohen, W. 2000. Data integration using similarity joins and a word-based information representation language. *ACM Trans. Inf. Syst. 18*, 3, 288–321.

Corley, C. and Mihalcea, R. 2005. Measures of text semantic similarity. In *Proceedings of the ACL workshop on Empirical Modeling of Semantic Equivalence* (Ann Arbor, MI).

Dolan, W., Quirk, C., and Brockett, C. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Erkan, G. and Radev, D. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Research 22*, 457–479.

Foltz, P., Kintsch, W., and Landauer, T. 1998. The measurement of textual coherence with latent semantic analysis. *Disc. Proc. 25*, 2–3, 285–307.

Frawley, W. 1992. *Linguistic Semantics*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Hatzivassiloglou, V., Klavans, J., and Eskin, E. 1999. Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. 203–212.

Islam, A. and Inkpen, D. 2006. Second order co-occurrence PMI for determining the semantic similarity of words. In *Proceedings of the International Conference on Language Resources and Evaluation*. (Genoa, Italy). 1033–1038.

Islam, A., Inkpen, D. Z., and Kiringa, I. 2008. Applications of corpus-based semantic similarity and word segmentation to database schema matching. *The VLDB Journal (Published online)*.

Jackendoff, R. 1983. *Semantics and Cognition*. MIT Press, Cambridge, MA.

Jarmasz, M. and Szpakowicz, S. 2003. Roget's thesaurus and semantic similarity. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. 212–219.

Jiang, J. and Conrath, D. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*.

[14]http://www.dmoz.org/

Katarzyna, W.-W. and Szczepaniak, P. 2005. Classification of rss-formatted documents using full text similarity measures. In *Proceedings of the 5th International Conference on Web Engineering*, D. Lowe and M. Gaedke, Eds. LNCS 3579. Springer, 400–405.

Ko, Y., Park, J., and Seo, J. 2004. Improving text categorization using the importance of sentences. *Inf. Proc. Manage. 40*, 65–79.

Kondrak, G. 2005. N-gram similarity and distance. In *Proceedings of the 12h International Conference on String Processing and Information Retrieval* (Buenos Aires, Argentina). 115–126.

Landauer, T. and Dumais, S. 1997. A solution to platos problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psych. Rev. 104*, 2, 211–240.

Landauer, T., Foltz, P., and Laham, D. 1998. Introduction to latent semantic analysis. *Dis. Proc. 25*, 2–3, 259–284.

Lapata, M. and Barzilay, R. 2005. Automatic evaluation of text coherence: Models and representations. In *Proceedings of the 19th International Joint Conference on AI*.

Leacock, C. and Chodorow, M. 1998. *WordNet: An electronic lexical database*. MIT Press, Chapter Combining local context andWordNet similarity for word sense identification, 265–283.

Lesk, M. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the SIGDOC Conference*.

Li, Y., Bandar, Z., and Mclean, D. 2003. An approach for measuring semantic similarity using multiple information sources. *IEEE Trans. Knowl. Data Eng. 15*, 4, 871–882.

Li, Y., McLean, D., Bandar, Z., O'Shea, J., and Crockett, K. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Trans. Knowl. Data Eng. 18*, 8, 1138–1149.

Lin, C. and Hovy, E. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the Human Language Technology Conference*.

Lin, D. 1998. An information-theoretic definition of similarity. In *Proceedings of the International Conference on Machine Learning*.

Liu, T. and Guo, J. 2005. Text similarity computing based on standard deviation. In *Proceedings of the International Conference on Intelligent Computing*, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds. Lecture Notes in Computer Science, vol. 3644. Springer-Verlag, New York, 456–464.

Liu, Y. and Zong, C. 2004. Example-based chinese-english mt. In *Proceedings of the 2004 IEEE International Conference on Systems, Man, and Cybernetics*. Vol. 1–7. IEEE Computer Society Press, Los Alamitos, CA, 6093–6096.

Madhavan, J., Bernstein, P., Doan, A., and Halevy, A. 2005. Corpus-based schema matching. In *Proceedings of the International Conference on Data Engineering*.

Maguitman, A., Menczer, F., Roinestad, H., and Vespignani, A. 2005. Algorithmic detection of semantic similarity. In *Proceedings of the 14th International World Wide Web Conference*.

Meadow, C., Boyce, B., and Kraft, D. 2000. *Text Information Retrieval Systems*, second ed. Academic Press.

Melamed, I. D. 1999. Bitext maps and alignment via pattern recognition. *Computat. Linguist. 25*, 1, 107–130.

Mihalcea, R., Corley, C., and Strapparava, C. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the American Association for Artificial Intelligence*. (Boston, MA).

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. 1993. Introduction to wordnet: An on-line lexical database. Tech. Rep. 43, Cognitive Science Laboratory, Princeton University, Princeton, NJ.

Miller, G. A. and Charles, W. G. 1991. Contextual correlates of semantic similarity. *Lang. and Cognitive Processes 6*, 1, 1–28.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting Association for Computational Linguistics*.

Park, E., Ra, D., and Jang, M. 2005. Techniques for improving web retrieval effectiveness. *Inf. Processing and Management 41*, 5, 1207–1223.

Resnik, P. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on AI*.

Rodriguez, M. A. and Egenhofer, M. J. 2003. Determining semantic similarity among entity classes from different ontologies. *IEEE Trans. Knowl. Data Eng. 15*, 2, 442–456.

Rubenstein, H. and Goodenough, J. B. 1965. Contextual correlates of synonymy. *Comm. ACM 8*, 10, 627–633.

Salton, G. and Lesk, M. 1971. *Computer Evaluation of Indexing and Text Processing*. Prentice Hall, Inc. Englewood Cliffs, NJ.

Schallehn, E., Sattler, K., and Saake, G. 2004. Efficient similarity-based operations for data integration. *Data Knowl. Eng. 48*, 361–387.

Schutze, H. 1998. Automatic word sense discrimination. *Computat. Linguist. 24*, 1, 97–124.

Sinclair, J., Ed. 2001. *Collins Cobuild English Dictionary for Advanced Learners*, third ed. Harper Collins.

Turney, P. 2001. Mining the web for synonyms: Pmi-ir versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning*.

Weeds, J., Weir, D., and McCarthy, D. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics*. 1015–1021.

Wiemer-Hastings, P. 2000. Adding syntactic information to lsa. In *Proceedings of the 22nd Annual Conference Cognitive Science Society*. 989–993.

Wu, Z. and Palmer, M. 1994. Verb semantics and lexical selection. In *Proceedings of the Annual Meeting Association for Computational Linguistics*.

Received May 2007; revised April 2008; accepted May 2008