Sentence Similarity Based on Semantic Nets and Corpus Statistics

Yuhua Li[1], David McLean[2], Zuhair Bandar[2], James D. O'Shea[2], Keeley Crockett[2]

1. School of Computing & Intelligent Systems

University of Ulster

Londonderry BT48 7JL, UK

Email: y.li@ulster.ac.uk

2. Department of Computing & Mathematics

Manchester Metropolitan University

Manchester M1 5GD, UK

Email: {d.mclean, z.bandar, j.d.oshea, k.crockett}@mmu.ac.uk

Abstract: Sentence similarity measures play an increasingly important role in text-related research and applications in areas such as text mining, web page retrieval and dialogue systems. Existing methods for computing sentence similarity have been adopted from approaches used for long text documents. These methods process sentences in a very high dimensional space and are consequently inefficient, require human input and are not adaptable to some application domains. This paper focuses directly on computing the similarity between very short texts of sentence length. It presents an algorithm that takes account of semantic information and word order information implied in the sentences. The semantic similarity of two sentences is calculated using information from a structured lexical database and from corpus statistics. The use of a lexical database enables our method to model human common sense knowledge and the incorporation of corpus statistics allows our method to be adaptable to different domains. The proposed method can be used in a variety of applications that involve text knowledge representation and discovery. Experiments on two sets of selected sentence pairs demonstrate that the proposed method provides a similarity measure that shows a significant correlation to human intuition.

Keywords: sentence similarity, semantic nets, corpus, natural language processing, word similarity

1 Introduction

Recent applications of natural language processing present a need for an effective method to compute the similarity between very short texts or sentences [25]. An example of this is a conversational agent/dialogue system with script strategies [1] in which sentence similarity is essential to the implementation. The employment of sentence similarity can significantly simplify the agent's knowledge base by using natural sentences rather than structural patterns of sentences. Sentence similarity will have internet related applications as well. In web page retrieval, sentence similarity has proved to be one of the best techniques for improving retrieval effectiveness, where titles are used to represent documents in the named page finding task [29]. 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 [8]. In text mining, sentence similarity is used as a criterion to discover unseen knowledge from textual databases [2]. In addition, the incorporation of short-text similarity is beneficial to applications such as text summarization [9], text categorization [15] and machine translation [21]. These exemplar applications show that the computing of sentence similarity has become a generic component for the research community involved in text-related knowledge representation and discovery.

Traditionally, techniques for detecting similarity between long texts (documents) have centred on analysing shared words [36]. Such methods are usually effective when dealing with long texts because similar long texts will usually contain a degree of co-occurring words. However, in short texts word co-occurrence may be rare or even null. This is mainly due to the inherent flexibility of natural language enabling people to express similar meanings using quite different sentences in terms of structure and word content. Since such surface information in short texts is very limited, this problem poses

a difficult computational challenge. The focus of this paper is on computing the similarity between very short texts, primarily of sentence length.

Although sentence similarity is increasingly in demand from a variety of applications as described earlier in this paper, the adaptation of available measures to computing sentence similarity has three major drawbacks. Firstly a sentence is represented in a very high dimensional space with hundreds or thousands of dimensions [18], [36]. This results in a very sparse sentence vector which is consequently computationally inefficient. High dimensionality and high sparsity can also lead to unacceptable performance in similarity computation [5]. Secondly some methods require the user's intensive involvement to manually pre-process sentence information [22]. Thirdly once the similarity method is designed for an application domain, it cannot be adapted easily to other domains. This lack of adaptability does not correspond to human language usage as sentence meaning may change, to varying extents, from domain to domain. To address these drawbacks, this paper aims to develop a method that can be used generally in applications requiring sentence similarity computation. An effective method is expected to be dynamic in only focusing on the sentences of concern, fully automatic without requiring users' manual work and readily adaptable across the range of potential application domains.

The next section reviews some related work briefly. Section 3 presents a new method for measuring sentence similarity. Section 4 provides implementation considerations related to obtaining information from knowledge bases. Section 5 shows the similarities calculated for a set of Natural Language Processing (NLP) related sentence pairs and carries out an experiment involving 32 human participants providing similarity ratings for a dataset of 30 selected sentence pairs. These results are then used to evaluate our similarity method. Section 5 concludes that the proposed method coincides with human perceptions about sentence similarity. Finally section 6 summarizes the work, draws some conclusions and proposes future related work.

2 Related Work

In general, there is extensive literature on measuring the similarity between documents or long texts [1], [12], [17], [24], but there are very few publications relating to the measurement of similarity between very short texts [10] or sentences. This section reviews some related work in order to explore the strengths and limitations of previous methods, and to identify the particular difficulties in computing sentence similarity. Related works can roughly be classified into three major categories: word co-occurrence methods, corpus-based methods, descriptive features-based methods.

The word co-occurrence methods are often known as the 'bag of words' method. It is commonly used in Information Retrieval (IR) systems [24]. The systems have a pre-compiled word list with *n* words. The value of *n* is generally in the thousands or hundreds of thousands in order to include all meaningful words in a natural language. Each document is represented using these words as a vector in *n*-dimensional space. A query is also considered as a document. The relevant documents are then retrieved based on the similarity between the query vector and the document vector. This technique relies on the assumption that more similar documents share more of the same words. If this technique were applied to sentence similarity, it would have three obvious drawbacks:

1) The sentence representation is not very efficient. The vector dimension *n* is very large compared to the number of words in a sentence, thus the resulting vectors would have many null components.
2) The word set in IR systems usually exclude function words such as *the*, *of*, *an*, etc. Function words are not very helpful for computing document similarity, but cannot be ignored for sentence similarity because they carry structural information, which is useful in interpreting sentence meaning. If function words were included, the value for *n* would be greater still.

3) Sentences with similar meaning do not necessarily share many words.

One extension of word co-occurrence methods is the use of a lexical dictionary to compute the similarity of a pair of words taken from the two sentences that are being compared (where one word is taken from each sentence to form a pair). Sentence similarity is simply obtained by aggregating similarity values of all word pairs [28]. Another extension of word co-occurrence techniques leads to the pattern matching methods which are commonly used in conversational agents and text mining [7]. Pattern matching differs from pure word co-occurrence methods by incorporating local structural information about words in the predicated sentences. A meaning is conveyed in a limited set of patterns where each is represented using a regular expression [14] (generally consisting of parts of words and various wildcards) to provide generalisation. Similarity is calculated using a simple pattern matching algorithm. This technique requires a complete pattern set for each meaning, in order to avoid ambiguity and mismatches. Manual compilation is an immensely arduous and tedious task. At present it is not possible to prove that a pattern set is complete and thus there is no automatic method for compiling such a pattern set. Finally, once the pattern sets are defined, the algorithm is unable to cope with unplanned novel utterances from human users.

One recent active field of research that contributes to sentence similarity computation is the methods based on statistical information of words in a huge corpus. Well known methods in corpus-based similarity are the latent semantic analysis (LSA) [10], [17], [18] and the Hyperspace Analogues to Language (HAL) model [5]. Some leading researchers in LSA boldly claim that LSA is a complete model of language understanding [17]. In LSA, a set of representative words needs to be identified from a large number of contexts (each described by a corpus). A *word by context* matrix is formed based on the presence of words in contexts. The matrix is decomposed by singular value decomposition (SVD) into the product of three other matrices including the diagonal matrix of singular values [19]. The diagonal singular matrix is truncated by

deleting small singular values. In this way, the dimensionality is reduced. The original *word by context matrix* is then reconstructed from the reduced dimensional space. Through the process of decomposition and reconstruction, LSA acquires word knowledge that spreads in contexts. When LSA is used to compute sentence similarity, a vector for each sentence is formed in the reduced dimension space, similarity is then measured by computing the similarity of these two vectors [10]. Because of the computational limit of SVD, the dimension size of the *word by context* matrix is limited to the several hundreds. As the input sentences may be from an unconstrained domain (and thus not represented in the contexts) some important words from the input sentences may not be included in the LSA dimension space. Secondly, the dimension is fixed and so the vector is fixed and is thus likely to be a very sparse representation of a short text such as a sentence. Like other methods, LSA ignores any syntactic information from the two sentences being compared and is understood to be more appropriate for larger texts than the sentences dealt with in this work [18].

Another important work in corpus-based methods is Hyperspace Analogues to Language (HAL) [5]. Indeed HAL is closely related to LSA and they both capture the meaning of a word or text using lexical co-occurrence information. Unlike LSA that builds an information matrix of words by text units of paragraphs or documents, HAL builds a word-by-word matrix based on word co-occurrences within a moving window of a pre-defined width. The window (typically with a width of 10 words) moves over the entire text of the corpus. An $N \times N$ matrix is formed for a given vocabulary of N words. Each entry of the matrix records the (weighted) word co-occurrences within the window moving through the entire corpus. The meaning of a word is then represented as a $2N$ dimensional vector by combining the corresponding row and column in the matrix. 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 so promising as LSA in the computation of similarity for short texts [5]. 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 large number of words are added to it.

The third category of related work is the descriptive features-based methods. The feature vector method tries to represent a sentence using a set of predefined features [22]. Basically a word in a sentence is represented using semantic features, for example, nouns may have features such as HUMAN (with value of human or nonhuman), SOFTNESS (soft or hard), and POINTNESS (pointed or rounded). A variation of feature vector methods is the introduction of primary features and composite features [12], [13]. Primary features are those primitive features that compare single items from each text unit. Composite features are the combination of pairs of primitive features. A text is then represented in a vector consisting of values of primary features and composite features. Similarity between two texts is obtained through a trained classifier. The difficulties for this method lie in the definition of effective features and in automatically obtaining values for features from a sentence. The preparation of a training vector set could be an impractical, tedious and time-consuming task. Moreover, features can be well-defined for concrete concepts; however it still is problematic to define features for abstract concepts.

Overall, the aforementioned methods compute similarity according to the co-occurring words in the texts, and ignore syntactic information. They work well for long texts because long texts have adequate information (i.e. they have a sufficient number of co-occurring words) for manipulation by a computational method. The proposed algorithm addresses the limitations of these existing methods by forming the word vector dynamically based entirely on the words in the compared sentences. The dimension of our vector is not fixed but varies with the sentence pair and so it is far

more computationally efficient than existing methods. Our algorithm also considers word order, which is a further aspect of primary syntactic information [1].

3 The Proposed Text Similarity Method

The proposed method derives text similarity from semantic and syntactic information contained in the compared texts. A text is considered to be a sequence of words each of which carries useful information. The words along with their combination structure make a text convey a specific meaning. Texts considered in this paper are assumed to be of sentence length.

Figure 1 shows the procedure for computing the sentence similarity between two candidate sentences. Unlike existing methods that use a fixed set of vocabulary, the proposed method dynamically forms a joint word set only using all the distinct words in the pair of sentences. For each sentence, a raw semantic vector is derived with the assistance of a lexical database. 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.

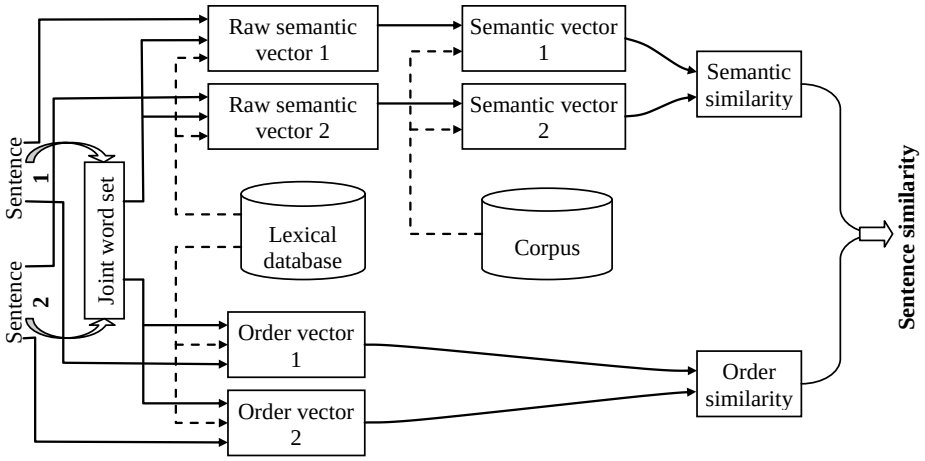

Figure 1. Sentence similarity computation diagram.

The following sub-sections present a detailed description of each of the above steps. Since semantic similarity between words is used both in deriving sentence semantic similarity and word order similarity, we will first describe our method for measuring word semantic similarity.

3.1 Semantic Similarity between Words

A number of semantic similarity methods have been developed in the previous decade. Different similarity methods have proved to be useful in some specific applications of computational intelligence [4], [23]. Generally these methods can be categorised into two groups: edge counting based (or dictionary/thesaurus based) methods and information theory based (or corpus based) methods, a detailed review on word similarity can be found in [20], [34]. After extensively investigating a number of methods, we proposed a word similarity measure which provides the best correlation to human judges for a benchmark word set as reported in [20]. This section summarises these research findings.

Thanks to the success of a number of computational linguistic projects, semantic knowledge bases are readily available, some examples being, WordNet [26], Spatial Date Transfer Standard [39] and Gene Ontology [38]. The knowledge bases tend to consist of a hierarchical structure modelling human common sense knowledge for a

particular domain, or in this case general English Language usage (WordNet [26]). The hierarchical structure of the knowledge base is important in determining the semantic distance between words (see Figure 2 for an example portion).

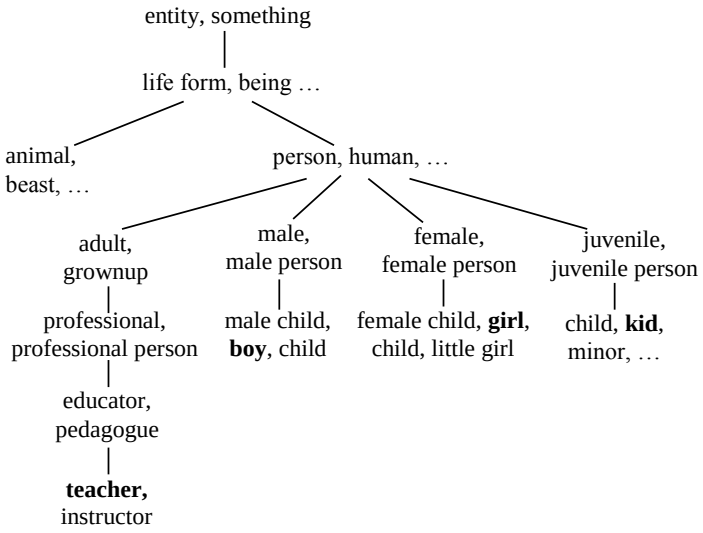

Figure 2. Hierarchical semantic knowledge base.

Given two words: w_1 and w_2, we need to find the semantic similarity $s(w_1, w_2)$. We can do this by analysis of the lexical knowledge base (in this paper we have used WordNet) as follows. Words are organised into synonym sets (synsets) in the knowledge base [26], with semantics and relation pointers to other synsets. Therefore we can find the first class in the hierarchical semantic network that subsumes the compared words. One direct method for similarity calculation is to find the minimum length of path connecting the two words [30]. For example, the shortest path between *boy* and *girl* in Figure 2 is *boy-male-**person**-female-girl*, the minimum path length is 4, the synset of ***person*** is called the subsumer for words of *boy* and *girl*; while the minimum path length between *boy* and *teacher* is 6. Thus we could say *girl* is more similar to *boy* than *teacher* to *boy*. Rada et al [30] demonstrated that this method works well on their much constrained medical semantic nets (with 15000 medical terms). However this method may be less accurate if it is applied to larger and more general semantic nets such as WordNet [26]. For example, the minimum length from *boy* to *animal* is 4, less than

from *boy* to *teacher*, but intuitively *boy* is more similar to *teacher* than to *animal* (unless you are cursing the boy). To address this weakness, the direct path length method must be modified by utilising more information from the hierarchical semantic nets. It is apparent that words at upper layers of the hierarchy have more general semantics and less similarity between them, while words at lower layers have more concrete semantics and more similarity. Therefore the depth of word in the hierarchy should be taken into account. In summary, similarity between words is determined not only by path lengths but also by depth. We propose that the similarity $s(w_1, w_2)$ between words w_1 and w_2 is a function of path length and depth as follows:

$$s(w_1, w_2) = f(l, h) \quad (1)$$

where, l is the shortest path length between w_1 and w_2, h is the depth of subsumer in the hierarchical semantic nets.

We assume that Equation (1) can be rewritten using two independent functions as:

$$s(w_1, w_2) = f_1(l) \cdot f_2(h) \quad (2)$$

f_1 and f_2 are transfer functions of path length and depth respectively. We call these information sources, of path length and depth, attributes.

3.1.1 Properties of Transfer Functions

Values of an attribute in Equation (2) may cover a large range up to infinity, while the interval of similarity should be finite with extremes of *exactly the same* to *no similarity at all*. If we assign exactly the same with a value of 1 and no similarity as 0, then the interval of similarity is [0, 1]. The direct use of information sources as a metric of similarity is inappropriate due to its infinite property. Therefore it is intuitive that the transfer function from information sources to semantic similarity is a non-linear function. Taking path length as an example, when the path length decreases to zero, the similarity would monotonically increase towards the limit 1, while path length increases

infinitely (although this would not happen in an organised lexical database), the similarity should monotonically decrease to 0. Therefore, to meet these constraints the transfer function must be a non-linear function. The non-linearity of the transfer function is taken into account in the derivation of the formula for semantic similarity between two words as in the following sub-sections.

3.1.2 Contribution of Path Length

For a semantic net hierarchy, as in Figure 2, the path length between two words, w_1 and w_2, can be determined from one of three cases:

1. w_1 and w_2 are in the same synset
2. w_1 and w_2 are not in the same synset, but the synset for w_1 and w_2 contain one or more common words. For example, in Figure 2, the synset for *boy* and synset for *girl* contain one common word *child.*
3. w_1 and w_2 are neither in the same synset nor do their synsets contain any common words.

Case 1 implies that w_1 and w_2 have the same meaning, we assign the semantic path length between w_1 and w_2 to 0. Case 2 indicates that w_1 and w_2 partially share the same features, we assign the semantic path length between w_1 and w_2 to 1. For case 3, we count the actual path length between w_1 and w_2. Taking the above considerations into account, we set $f_1(l)$ in Equation (2) to be a monotonically decreasing function of l:

$$f_1(l) = \mathrm{e}^{-\alpha l} \qquad (3)$$

where α is a constant. The selection of the function in exponential form ensures that f_1 satisfies the constraints discussed in Section 3.2.1, and the value of f_1 is within the range from 0 to 1.

3.1.3 Scaling Depth Effect

Words at upper layers of hierarchical semantic nets have more general concepts and less semantic similarity between words than words at lower layers. This behaviour must be taken into account in calculating $s(w_1, w_2)$. We therefore need to scale down $s(w_1, w_2)$ for subsuming words at upper layers and to scale up $s(w_1, w_2)$ for subsuming words at lower layers. As a result, $f_2(h)$ should be a monotonically increasing function with respect to depth h. We set f_2 as:

$$f_2(h) = \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \quad (4)$$

where $\beta > 0$ is a smoothing factor. As $\beta \to \infty$, then the depth of a word in the semantic nets is not considered.

In summary, we propose a formula for a word similarity measure as:

$$s(w_1, w_2) = e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \quad (5)$$

where $\alpha \in [0,1], \beta \in (0,1]$ are parameters scaling the contribution of shortest path length and depth, respectively. The optimal value of α and β are dependant on the knowledge base used and can be determined using a set of word pairs with human similarity ratings. For WordNet, the optimal parameters for the proposed measure are: α=0.2, β=0.45 as reported in [20].

3.2 Semantic Similarity between Sentences

Sentences are made up of words, so it is reasonable to represent a sentence using the words in the sentence. Unlike classical methods that use a pre-compiled word list containing hundreds of thousands of words, our method dynamically forms the semantic vectors solely based on the compared sentences. Recent research achievements in semantic analysis are also adapted to derive an efficient semantic vector for a sentence.

Given two sentences: T_1 and T_2, a joint word set is formed:

$$T = T_1 \cup T_2$$
$$= \{w_1 \quad w_2 \quad \cdots \quad w_m\}$$

The joint word set T contains all the distinct words from T_1 and T_2. Since inflectional morphology may cause a word to appear in a sentence with different forms that convey a specific meaning for a specific context, we use word form as it appears in the sentence. For example, *boy* and *boys*, *woman* and *women* are considered as four distinct words and all included in the joint word set. Thus the joint word set for two sentences

T_1: RAM keeps things being worked with.

T_2: The CPU uses RAM as a short-term memory store.

is

T = {RAM keeps things being worked with The CPU uses as a short-term memory store}

Since the joint word set is purely derived from the compared sentences, it is compact with no redundant information. The joint word set, T, can be viewed as the semantic information for the compared sentences. Each sentence is readily represented by the use of the joint word set as follows. The vector derived from the joint word set is called the lexical semantic vector, denoted by $\check{\boldsymbol{s}}$. Each entry of the semantic vector corresponds to a word in the joint word set, so the dimension equals the number of words in the joint word set. The value of an entry of the lexical semantic vector, $\check{s}_i(i=1,2,...,m)$, is determined by the semantic similarity of the corresponding word to a word in the sentence. Take T_1 as an example:

Case 1: If w_i appears in the sentence, $\check{s}_i$ is set to 1.

Case 2: If w_i is not contained in T_1, a semantic similarity score is computed between w_i and each word in the sentence T_1, using the method presented in Section 3.1. Thus the most similar word in T_1 to w_i is that with the highest similarity score ς. If ς exceeds a preset threshold, then $\check{s}_i = \varsigma$, otherwise $\check{s}_i = 0$.

The reason for the introduction of the threshold is two fold. Firstly, since we use the word similarity of distinct words (different words) the maximum similarity scores

may be very low, indicating that the words are highly dissimilar. In this case we would not want to introduce such noise to the semantic vector. Secondly classical word matching methods [1] can be unified into the proposed method by simply setting the threshold equal to one. Unlike classical methods, we also keep all function words. This is because function words carry syntactic information that cannot be ignored if a text is very short (e.g. sentence length). Although function words are retained in the joint word set, they contribute less to the meaning of a sentence than other words. Furthermore different words contribute towards the meaning of a sentence to differing degrees. Thus a scheme is needed to weight each word. We weight the significance of a word using its information content [32].

It has been shown that words that occur with a higher frequency (in a corpus) contain less information than those that occur with lower frequencies [24]. The information content of a word is derived from its probability in a corpus (see Section 4.2.2 for details). Each cell is weighted by the associated information $I(w_i)$ and $I(\tilde{w}_i)$. Finally the value of an entry of the semantic vector is:

$$s_i = \breve{s} \cdot I(w_i) \cdot I(\tilde{w}_i) \qquad (6)$$

where w_i is a word in the joint word set, $\tilde{w}_i$ is its associated word in the sentence. The use of $I(w_i)$ and $I(\tilde{w}_i)$ allows the concerned two words contribute to similarity based on their individual information contents. The semantic similarity between two sentences is defined as the cosine coefficient between the two vectors:

$$S_s = \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{\|\mathbf{s}_1\| \cdot \|\mathbf{s}_2\|} \qquad (7)$$

It is worth noting that the proposed method does not currently conduct word sense disambiguation for polysemous words. This is based on the following considerations. Firstly we wanted our model to be as simple as possible and not too demanding in terms of computing resources. The integration of word sense disambiguation would scale up the model complexity. Secondly existing sentence similarity methods have not included

word sense disambiguation. This might be a consequence of the first factor. Thirdly, even though the proposed method does not use disambiguation, it still performs well, achieving promising results as shown later in our experiments.

3.3 Word Order Similarity between Sentences

Let us consider a pair of sentences, T_1 and T_2, that contain exactly the same words in the same order with the exception of two words from T_1 which occur in the reverse order in T_2. For example:

T_1: A quick brown dog jumps over the lazy fox.

T_2: A quick brown fox jumps over the lazy dog.

Since these two sentences contain the same words, any methods based on "bag of words" will give a decision that T_1 and T_2 are exactly the same. However it is clear for a human interpreter that T_1 and T_2 are only similar to some extent. The dissimilarity between T_1 and T_2 is the result of the different word order. Therefore a computational method for sentence similarity should take into account the impact of word order.

For the example pair of sentences T_1 and T_2, the joint word set is:

T = {A quick brown dog jumps over the lazy fox}

We assign a unique index number for each word in T_1 and T_2. The index number is simply the order number that the word appears in the sentence. For example, the index number is 4 for *dog* and 6 for *over* in T_1. In computing the word order similarity, a word order vector, **r**, is formed for T_1 and T_2 respectively based on the joint word set T. Taking T_1 as an example, for each word w_i in T we try to find the same or the most similar word in T_1 as follows.

1. If the same word is present in T_1, we fill the entry for this word in $\mathbf{r}_1$ with the corresponding index number from T_1. Otherwise, we try to find the most similar word $\tilde{w}_i$ in T_1 (as described in section 3.2)
2. If the similarity between w_i and $\tilde{w}_i$ is greater than a pre-set threshold, the entry of w_i in $\mathbf{r}_1$ is filled with the index number of $\tilde{w}_i$ in T_1.

3. If the above two searches fail, the entry of w_i in $\mathbf{r}_1$ is 0.

Having applied the above procedure, the word order vectors for T_1 and T_2 are $\mathbf{r}_1$ and $\mathbf{r}_2$ respectively. For the example sentence pair, we have:

$\mathbf{r}_1$ = {1 2 3 4 5 6 7 8 9}

$\mathbf{r}_2$ = {1 2 3 9 5 6 7 8 4}

Thus a word order vector is the basic structural information carried by a sentence. The task of dealing with word order is then to measure how similar the word order in two sentences is. We propose a measure for measuring word order similarity of two sentences as:

$$S_r = 1 - \frac{\|\mathbf{r}_1 - \mathbf{r}_2\|}{\|\mathbf{r}_1 + \mathbf{r}_2\|} \qquad (8)$$

That is, word order similarity is determined by the normalised difference of word order. The following analysis will demonstrate that S_r is an efficient metric for indicating word order similarity. To simplify the analysis, we will consider only a single word order difference, as in sentences T_1 and T_2.

Given two sentences: T_1 and T_2, where both sentences contain exactly the same words and the only difference is that a pair of words in T_1 appears in the reverse order in T_2. The word order vectors are:

$\mathbf{r}_1 = \{a_1 \cdots a_j \cdots a_{j+k} \cdots a_m\}$ for T_1

$\mathbf{r}_2 = \{b_1 \cdots b_j \cdots b_{j+k} \cdots b_m\}$ for T_2

a_j and a_{j+k} are the entries for the considered word pair in T_1 , b_j and b_{j+k} are the corresponding entries for the word pair in T_2 , k is the number of words from w_j to w_{j+k} . From the above assumptions, we have:

$a_i = b_i = i$ for i=1, 2, …, m except $i \neq j, j+k$

$a_j = b_{j+k} = j$

$a_{j+k} = b_j = j+k$

$\|\mathbf{r}_1\| = \|\mathbf{r}_2\| \equiv \|\mathbf{r}\|$

then:

$$S_r = 1 - \frac{k}{\sqrt{2\|\mathbf{r}\|^2 - k^2}} \qquad (9)$$

We can also derive the same formula for a sentence pair with only one different word at the *k*th entry. For the more general case with a more significant difference in word order or a larger number of different words, the analytical form of the proposed metric becomes more complicated (which we do not intend to present in this paper). The above analysis shows that S_r is a suitable indication of word order information. S_r equals 1 if there is no word order difference. S_r is greater or equal to 0 if word order difference is present. Since S_r is a function of k, it can reflect the word order difference and the compactness of a word pair. The following features of the proposed word order metric can also be observed.

1. S_r can reflect the words shared by two sentences.
2. S_r can reflect the order of a pair of the same words in two sentences. It only indicates the word order, while it is invariant regardless of the location of the word pair in an individual sentence.
3. S_r is sensitive to the distance between the two words of the word pair. Its value decreases as the distance increase.
4. For the same number of different words or the same number of word pairs in a different order, S_r is proportional to the sentence length (number of words), its value increases as the sentence length increases. This coincides with intuitive knowledge, that is, two sentences would share more of the same words for a certain number of different words or different word order if the sentence length is longer.

Therefore the proposed metric is a good one for indicating the word order in terms of word sequence and location in a sentence.

3.4 Overall Sentence Similarity

Semantic similarity represents the lexical similarity. On the other hand, word order similarity provides information about the relationship between words: which words appear in the sentence and which words come before or after other words. Both semantic and syntactic information (in terms of word order) play a role in conveying the meaning of sentences. Thus the overall sentence similarity is defined as a combination of semantic similarity and word order similarity:

$$\begin{aligned} S(T_1, T_2) &= \delta S_s + (1-\delta) S_r \\ &= \delta \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{\|\mathbf{s}_1\| \cdot \|\mathbf{s}_2\|} + (1-\delta) \frac{\|\mathbf{r}_1 - \mathbf{r}_2\|}{\|\mathbf{r}_1 + \mathbf{r}_2\|} \end{aligned} \qquad (10)$$

where $\delta \leq 1$ decides the relative contributions of semantic and word order information to the overall similarity computation. Since syntax plays a subordinate role for semantic processing of text [11], δ should be a value greater than 0.5, i.e., $\delta \in (0.5,1]$.

4 Implementation Using Semantic Nets and Corpus Statistics

Two databases were used in the implementation of the proposed method, namely WordNet [26] and the Brown Corpus [3]. This section provides a brief description of these two databases and then presents the search in the lexical taxonomy and the derivation of statistics from the corpus.

4.1 The Databases

WordNet is an on-line semantic dictionary - a lexical database, developed at Princeton by a group led by Miller [26]. The version used in this study is WordNet 1.6 which has 121,962 words organised in 99,642 synonym sets. WordNet partitions the lexicon into nouns, verbs, adjectives, and adverbs. These sets of words are organised into synonym sets, called synsets. A synset represents a concept in which all words have a similar meaning. Thus words in a synset are interchangeable in some syntax. Knowledge in a synset includes the definition of these words as well as pointers to other related synsets.

The Brown Corpus [3] comprises of 1,014,000 American English words and was compiled at the Brown University for standard texts in 1961.
In this study, WordNet is the main semantic knowledge base for the calculation of semantic similarity, while the Brown Corpus is used to provide information content.

4.2 Obtaining Information Sources

The implementation of semantic similarity measures consists of two sub-tasks concerning preparation of the information sources that are used in the formation of the semantic and word order vectors. Firstly, a search of the semantic net is performed for the shortest path length between the synsets containing the compared words and the depth of the first synset subsuming the synsets corresponding to the compared words [20]. Secondly, the calculation of the necessary statistical information from the Brown Corpus is performed.

4.2.1 Search in WordNet

Synsets in WordNet are designed in a tree-like hierarchical structure ranging from many specific terms at the lower levels to a few generic terms at the top. The lexical hierarchy is connected by following trails of superordinate terms in “is a” or “is a kind of” (ISA) relations. To establish a path between two words, each climbs up the lexical tree until the two climbing paths meet. The synset at the meeting point of the two climbing paths is called the subsumer, a path connecting the two words is then found through the subsumer. Path length is obtained by counting synset links along the path between the two words. The depth of the subsumer is derived by counting the levels from the subsumer to the top of the lexical hierarchy. If a word is polysemous (i.e., a word having many meanings), multiple paths may exist between the two words. Only the shortest path is then used in calculating semantic similarity between words. The subsumer on the shortest path is considered in deriving the depth of the subsumer. Most previous similarity measures only use the shortest path length from this ISA search. It is

commonly accepted that other semantic relations also contribute to the determination of semantic similarity. One important such relation is part-whole (or HASA) relation. Thus we also search for HASA relations in WordNet in obtaining the shortest path length as did [20], [34]. In addition, a mechanism is used to deal with the following exceptional case, i.e., words not contained in WordNet. If the word is not in WordNet, then the search will not proceed and the word similarity is simply assigned to zero. A warning message on validity of the similarity is prompted to the user. Alternatively, this problem could be solved if the missing word exists in another lexical database through knowledge fusion [34].

4.2.2 Statistics from the Brown Corpus

The probability of a word *w* in the corpus is computed simply as the relative frequency:

$$\hat{p}(w) = \frac{n+1}{N+1} \qquad (11)$$

where *N* is the total number of words in the corpus, *n* is the frequency of the word *w* in the corpus (increased by 1 to avoid presenting an undefined value to the logarithm). Information content of *w* in the corpus is defined as:

$$I(w) = -\frac{\log p(w)}{\log(N+1)} = 1 - \frac{\log(n+1)}{\log(N+1)} \qquad (12)$$

so $I \in [0,1]$.

4.3 Illustrative Example: Similarities for a selected sentence pair

To illustrate how to compute the overall sentence similarity for a pair of sentences, we provide below a detailed description of our method for two example sentences:

T_1: RAM keeps things being worked with.

T_2: The CPU uses RAM as a short-term memory store.

The joint word set is:

T = {RAM keeps things being worked with The CPU uses as a short-term memory store}

Semantic vectors for T_1 and T_2 can be formed from T and corpus statistics. The process of deriving semantic vectors for T_1 is shown in Table 1.

	RAM	keeps	things	being	worked	with	The	CPU	uses	as	a	short-term	memory	store
RAM	1												0.8147	0.8147
keeps		1												
things			1					0.2802	0.4433					
being				1										
worked					1									
with						1								
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
$\check{s}$	1	1	1	1	1	1	0	0.2802	0.4433	0	0	0	0.8147	0.8147
Weight	I(RAM) I(RAM)	I(keeps) I(keeps)	I(things) I(things)	I(being) I(being)	I(worked) I(worked)	I(with) I(with)	I(The) I(The)	I(CPU) I(things)	I(uses) I(things)	I(as) I(as)	I(a) I(a)	I(short-term) I(short-term)	I(memory) I(RAM)	I(store) I(RAM)

Table 1. The process for deriving the semantic vector.

In the table, the first row lists words in the joint word set T, the first column lists words in sentence T_1 and all words are listed in order as they appear in T and T_1. For each word in T, if the same word exists in T_1, the cell at the cross point is set to 1. Otherwise the cell at the cross point of the most similar word is set to their similarity value or 0, dependent on whether the highest similarity value exceeds the pre-set threshold which was set to 0.2[1] in our experiments. For example, the word *memory* is not in T_1, but the most similar word is *RAM*, with a similarity of 0.8147. Thus, the cell at the cross point of *memory* and *RAM* is set to 0.8147 as it exceeds the threshold of 0.2. All other cells are left empty. The lexical vector $\check{\boldsymbol{s}}$ is obtained by selecting the largest value in each column. The last row lists the corresponding information content for weighting the significance of the word. As a result, the semantic vector for T_1 is:

$\mathbf{s}_1$={0.390 0.330 0.179 0.146 0.239 0.074 0 0.082 0.1 0 0 0 0.263 0.288}

In the same way, we get:

[1] Empirically derived threshold, word similarity values of less than 0.2 are intuitively too dissimilar. This value may change for semantic nets other than WordNet.

$\mathbf{s}_2$={0.390 0 0.1 0 0 0 0.023 0.479 0.285 0.075 0.043 0.354 0.267 0.321}

From $\mathbf{s}_1$ and $\mathbf{s}_2$, the semantic similarity between the two sentences is S_s=0.6139. Similarly the word order vectors are derived as:

$\mathbf{r}_1$={1 2 3 4 5 6 0 3 3 0 0 0 1 1}

$\mathbf{r}_2$={4 0 3 0 0 0 1 2 3 5 6 7 8 9}

and thus S_r=0.2023.

Finally the similarity between sentences "RAM keeps things being worked with" and "The CPU uses RAM as a short-term memory store" is 0.5522, using 0.85 for δ[2]. This pair of sentences has only one co-occurrence word *RAM*, but the meaning of the sentences is similar. Word co-occurrence methods would result in a very low similarity measure [24], while the proposed method gives a relatively high similarity. This example demonstrates that the proposed method can capture the meaning of the sentence regardless of the co-occurrence of words.

5 Experimental Results

Although a few related studies have been published, there are currently no suitable benchmark datasets (or even standard text sets) for the evaluation of sentence (or very short text) similarity methods. Building such a dataset is not a trivial task due to subjectivity in the interpretation of language, which is in part due to the lack of deeper contextual information. Thus the construction of a suitable data set would require a large-scale psychological study over a cross-section of (the common) language speakers so as to include different cultural backgrounds. Such a large study is outside the scope of this paper but in order to evaluate our similarity measure a preliminary data set of sentence pairs was constructed with human similarity scores provided using 32 participants (this will form part of a larger future study). These sentences all consist of dictionary definitions of words and so a further dataset of non-definitive sentences was

[2] Empirically derived value through experiments on sentence pairs.

produced from the NLP literature. Currently no human similarities for this second dataset exist so it is left to the reader to judge our algorithm's performance for each of these sentence pairs.

Our similarity method requires three parameters to be determined before use: a threshold for deriving the semantic vector, a threshold for forming the word order vector, and a factor δ for weighting the significance between semantic information and syntactic information. All parameters in the following experiments were empirically found using a small set of sentence pairs, evidence from previous publications [20][11] and intuitive considerations as follows. Since syntax plays a subordinate role for semantic processing of text, we weighted the semantic part higher, 0.85 for δ. For the semantic threshold, we considered two aspects: to detect and utilise similar semantic characteristics of words to the greatest extent and to keep the noise low. This requires us to use a semantic threshold which is small, but not too small. Using a small threshold allows the model to capture sufficient semantic information distributed across all of the words. However too small a threshold will introduce excessive noise to the model causing a deterioration of the overall performance. A similar consideration applied to the word order threshold, but we used a higher value. For the word order vector to be useful the pair of linked words (the most similar words from the two sentences) must intuitively be quite similar, as the relative ordering of less similar pairs of words provides very little information. Based on these considerations, we first chose some starting values for the three parameters and then identified the appropriate values using the selected sentence pairs. In this way we empirically found 0.4 for word order threshold, 0.2 for semantic threshold and 0.85 for δ.

5.1 Selected NLP Sentences

Sentence pairs in Table 2, were selected from a variety of papers and books on natural language understanding. It can be seen that the similarities in the table are fairly consistent with human intuition. One obvious exception to this is the first pair of

sentences in which the word ‘bachelor’ has been replaced with a phrase ‘unmarried man’. As our technique compares words on a word-by-word basis, such multiple word phrases are currently missed, although similarities are found between the word pairs: *bachelor-man* and *bachelor-unmarried*. In addition, there is a big difference in similarity between examples 6 and 14, which only differ in the type of fruit involved (apple vs orange). This difference is the consequence of neglecting multiple senses of polysemous words as stated in Section 3.2. Orange is a colour as well as a fruit and is found more similar to another word on this basis. Word sense disambiguation may narrow this difference and it needs to be investigated in future work.

Sentence Pair	**Similarity**	**Sentence Pair**	**Similarity**
1. I like that bachelor. I like that unmarried man.	0.561	2. I have a pen. Where do you live?	0
3. John is very nice. Is John very nice?	0.977	4. Red alcoholic drink. A bottle of wine.	0.585
5. It is a dog. That must be your dog.	0.739	6. Red alcoholic drink. Fresh orange juice.	0.611
7. It is a dog. It is a log.	0.623	8. Red alcoholic drink. An English dictionary.	0
9. It is a dog. It is a pig.	0.790	10. Dogs are animals. They are common pets.	0.738
11. I have a hammer. Take some nails.	0.508	12. Canis familiaris are animals. Dogs are common pets.	0.362
13. I have a pen. Where is ink.	0.129	14. Red alcoholic drink. Fresh apple juice.	0.420
15. A glass of cider. A full cup of apple juice.	0.678	16. I have a hammer. Take some apples	0.121

Table 2. Similarities between selected sentence pairs.

5.2 Experiment with Human Similarities of Sentence Pairs

In order to evaluate our similarity measure, we 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. We began with 65 noun word pairs whose semantic similarity

was originally measured by Rubenstein and Goodenough [35]. This data has been used in many experiments in the intervening years, its properties are well-known and it has shown stability when re-rated with new groups of participants. The frequency distribution of the data exhibits a strong bias, however, with two-thirds of the data falling in the upper and lower quarters of the similarity range. A specific subset of 30 pairs has been used, which reduces bias in the frequency distribution [6], [27].

5.2.1 Materials

We began with the set of 65 noun pairs from Rubenstein & Goodenough and replaced them with their definitions from the Collins Cobuild dictionary [37]. Cobuild dictionary definitions are "…written in full sentences, using vocabulary and grammatical structures that occur naturally with the word being explained." The dictionary is constructed using information from a large corpus, the Bank of English, which contains 400 million words. Where more than one sense of a word was given we chose the first noun sense in the list. Two of the definitions were modified. The noun "Smile" was simply defined in terms of the verb "to smile." We substituted a phrase from the verb definition into the noun definition to form a usable sentence. There are some similar problems where one noun is defined in terms of another e.g. Automobile/Car, Cord/String, and Grin/Smile. As each of these combinations is used in the data set we have not made any substitutions in the definitions. The definition of "Bird" was split over three short sentences. We considered all to contribute to a distinctive definition so we combined them as phrases in a single, longer sentence.

Two of the word pairs have definitions that are genuinely virtually identical, Rooster/Cock and Midday/Noon. The complete sentence data set used in this study is available at *http://www.docm.mmu.ac.uk/STAFF/D.McLean/SentenceResults.htm*

5.2.2 Procedure

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 & Goodenough [35]. Each sentence pair was presented on a separate sheet. The order of presentation of the sentence pairs was randomised in each questionnaire. The order of the two sentences making up each pair was also randomised. This was to prevent any bias being introduced by order of presentation. The participants were asked to complete the questionnaire in their own time, and to work through from start to end in a single sitting. A rubric was provided which contained linguistic anchors for the five major scale points 0.0, 1.0, 2.0, 3.0, 4.0 - taken from a study by Charles [6]. This is important because, according to Charles it yields "psychometric properties analogous to an interval scale." It is common practice in similarity measurement to use statistics such as mean, standard deviation and Pearson product-moment correlation. All of these require the data to be measured on an interval scale or better. Use of the linguistic anchors reconciles these otherwise conflicting requirements.

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 towards the low similarity end of the scale. Following a similar procedure to Miller and Charles [27] 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. These can be seen in Table 3, all human similarity scores are provided as the mean score for each pair and have been scaled into the range [0..1], for comparison with our method's similarity measure (algorithm similarity measure).

R&G No.	R&G Word Pair	Human Similarity (Mean)	Algorithm Similarity Measure	R&G No.	R&G Word Pair	Human Similarity (Mean)	Algorithm Similarity Measure
1	Cord smile	0.01	0.33	51	Glass tumbler	0.14	0.65
5	Autograph shore	0.01	0.29	52	Grin smile	0.49	0.49
9	Asylum fruit	0.01	0.21	53	Serf slave	0.48	0.39
13	Boy rooster	0.11	0.53	54	Journey voyage	0.36	0.52
17	Coast forest	0.13	0.36	55	Autograph signature	0.41	0.55
21	Boy sage	0.04	0.51	56	Coast shore	0.59	0.76
25	Forest graveyard	0.07	0.55	57	Forest woodland	0.63	0.70
29	Bird woodland	0.01	0.33	58	Implement Tool	0.59	0.75
33	Hill woodland	0.15	0.59	59	Cock rooster	0.86	1.00
37	Magician oracle	0.13	0.44	60	Boy lad	0.58	0.66
41	Oracle sage	0.28	0.43	61	Cushion pillow	0.52	0.66
47	Furnace stove	0.35	0.72	62	Cemetery graveyard	0.77	0.73
48	Magician wizard	0.36	0.65	63	Automobile car	0.56	0.64
49	Hill mound	0.29	0.74	64	Midday noon	0.96	1.0
50	Cord string	0.47	0.68	65	Gem jewel	0.65	0.83

Table 3. Sentence data set results.

5.2.3 Results and Discussion

Our algorithm's similarity measure achieved a reasonably good Pearson correlation coefficient of 0.816 with the human ratings, significant at the 0.01 level . However, a further factor should be taken into consideration, what is the best performance that could be expected from an algorithmic measure under this particular set of experimental

conditions? An upper bound was set in a comparative study of word similarity techniques by calculating the correlations between individual participants and the group using leave-one-out resampling [32], then finding the mean. In a similar manner, we calculated the correlation coefficient (Correlation *r*) for the judgements of each participant against the rest of the group and then took the mean. The results are presented in Table 4.

	Correlation *r*	Comment
Algorithm Similarity Measure	0.816	With average of all participants, significant at 0.01 level
Mean of all participants	0.825	Standard Deviation 0.072
Worst participant	0.594	
Best participant	0.921	

Table 4. Similarity correlations.

If we take the performance of the typical human, 0.825 as the upper bound then it reasonable to say that our similarity measure is performing well at 0.816, within the constraints of the experiment.

Comparing the word-pair ratings from Rubenstein and Goodenough with the corresponding sentence-pair ratings from our technique (Table 3), it is apparent that people perceive the semantic similarities of words differently from their definitions. Inspection of the word-pair vs sentence-pair for the full data set reveals a clear and regular non-linear relationship, further discussion of which is beyond the scope of this paper.

It is worth giving some consideration to the skew in the frequency distribution of the data set. The Rubenstein and Goodenough data has a frequency bias towards the extremes (high and low ends of the similarity scale) of the word-pair data set and suggested that participants may react differently to numerically equal intervals on the similarity scale. It has been postulated in word similarity studies, that participants take an accommodating approach by selecting the most similar sense of a polysemous word, artificially inflating the semantic similarity rating for some word pairs. We argue that

sentences carry their own context with them, largely disambiguating any polysemous words they contain to specific senses. Evidence supporting this comes from the Glass/Tumbler pair. This was scored 3.45 as a word pair in the Rubenstein & Goodenough trials and 0.55 as a sentence pair in our trials. This is consistent with the word pair judges interpreting Glass as an item to drink out of, whereas the definition in the sentence pair is of the substance glass. Similarly the Magician/Wizard pair was scored 3.21 as a word pair and 1.42 as sentence pair, this is consistent with the word Magician being interpreted as a practitioner of magic, whereas the sentence definition covers the "conjurer" sense. Finally it is worth noting that the Cord/String, Automobile/Car and Grin/Smile pairs were rated about halfway between minimum and maximum similarity, indicating that participants did not automatically substitute the semantic content of the second definition into the first.

6 Conclusions

This paper presented a method for measuring the semantic similarity between sentences or very short texts, based on semantic and word order information. Firstly, semantic similarity is derived from a lexical knowledge base and a corpus. The lexical knowledge base models common human knowledge about words in a natural language, this knowledge is usually stable across a wide range of language application areas. A corpus reflects the actual usage of language and words. Thus our semantic similarity not only captures common human knowledge, but it is also able to adapt to an application area using a corpus specific to that application. Secondly, the proposed method considers the impact of word order on sentence meaning. The derived word order similarity measures the number of different words as well as the number of word pairs in a different order. The overall sentence similarity is then defined as a combination of semantic similarity and word order similarity. Considering the view that word order plays a subordinate role for interpreting sentence meaning, we weight word order similarity less in defining the overall sentence similarity. To evaluate our similarity algorithm, we collected a set of

sentence pairs from a variety of articles and books in computational linguistics. An initial experiment on this data illustrates that the proposed method provides similarity measures that are fairly consistent with human knowledge. Next we constructed a data set of 30 sentence pairs using a dictionary definition for each of the Rubenstein and Goodenough word pairs [35]. The sentences were rated by human participants as a benchmark for comparison with our method which performed well on this data set.

Further work will include the construction of a more varied sentence pair dataset with human ratings and an improvement to the algorithm to disambiguate word sense using the surrounding words to give a little contextual information. Currently comparison with some of the other algorithms discussed is very difficult due to a lack of any other published results on sentence similarities (a benchmark data set) and a variety of problems in re-implementing these algorithms for this domain. These include the substantial amount of parameters which must be manually set and the definition of features.

Acknowledgements

We would like to thank the three anonymous referees for their insightful comments to the improvement in technical contents and paper presentation.

References

[1] J. Allen, Natural Language Understanding. Benjamin Cummings, Redwood City, CA, 1995.

[2] J. Atkinson-Abutridy, C. Mellish, and S. Aitken, "Combining information extraction with genetic algorithms for text mining," IEEE Intelligent Systems, vol. 19, no. 3, 2004.

[3] Brown Corpus Information, http://clwww.essex.ac.uk/w3c/corpus_ling/content/corpora/list/private/brown/brown.html

[4] A. Budanitsky and G. Hirst, "Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures," Workshop WordNet and Other Lexical Resources, Second Meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, 2001.

[5] C. Burgess, K. Livesay, and K. Lund, "Explorations in context space: Words, sentences, discourse," Discourse Processes, vol. 25, no. 2-3, pp. 211-257, 1998.

[6] W.G. Charles, "Contextual correlates of meaning," Applied Psycholinguistics, vol. 21, no. 4, pp. 505-524, 2000.

[7] J. H. Chiang and H.C. Yu, "Literature extraction of protein functions using sentence pattern mining," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 8, pp. 1088-1098, 2005.

[8] T.A.S. Coelho, P.P. Calado, L.V. Souza, B. Ribeiro-Neto, and R. Muntz, "Image retrieval using multiple evidence ranking," IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 4, pp. 408-417, 2004.

[9] G. Erkan and D.R. Radev, "LexRank: Graph-based lexical centrality as salience in text summarization," Journal of Artificial Intelligence Research, vol. 22, pp. 457-479, 2004.

[10] P.W. Foltz, W. Kintsch, and T.K. Landauer, "The measurement of textual coherence with latent semantic analysis," Discourse Processes, vol. 25, no. 2-3, pp. 285-307, 1998.

[11] P. Wiemer-Hastings, "Adding syntactic information to LSA," Proceedings of the Twenty-second Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, Mahwah, NJ, pp. 989-993, 2000.

[12] V. Hatzivassiloglou, J. Klavans, and E. Eskin, "Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning," Joint SIGDAT Conference on Empirical Methods in NLP and Very Large Corpora, University of Maryland, College Park, MD, USA, 1999.

[13] V. Hatzivassiloglou, J. Klavans, and E. Eskin, "Detecting similarity by applying leaning over indicators," 37th Annual Meeting of the Association for Computational Linguistics, University of Maryland, 1999.

[14] D. Jurafsky and J.H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and speech Recognition. Prentice Hall, 2000.

[15] 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.

[16] H. Kozima, "Computing lexical cohesion as a tool for text analysis," Ph.D. Thesis, Course in Computer Science and Information Mathematics, Graduate School of Electro-Communications, University of Electro-Communications, 1994.

[17] T.K. Landauer, D. Laham, B. Rehder, and M.E. Schreiner, "How well can passage meaning be derived without using word order? A comparison of latent semantic analysis and humans," Proceedings of the 19th Annual Meeting of the Cognitive Science Society, Erlbaum, Mawhwah, NJ, pp. 412-417, 1997.

[18] T.K. Landauer, P.W. Foltz, and D. Laham, "Introduction to latent semantic analysis," Discourse Processes, vol. 25, no. 2-3, pp. 259-284, 1998.

[19] T.K. Landauer, D. Laham, and P.W. Foltz, "Learning human-like knowledge by singular value decomposition: A progress report," M.I. Jordan, M.J. Kearns, S.A. Solla (Eds.), Advances in Neural Information Processing Systems 10, MIT Press, Cambridge, pp. 45-51, 1998.

[20] Y.H. Li, Z. Bandar, and D. McLean, "An approach for measuring semantic similarity using multiple information sources," IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 4, pp. 871-882, 2003.

[21] Y. Liu and C.Q. Zong, "Example-based Chinese-English MT," 2004 IEEE International Conference on Systems, Man & Cybernetics, vol. 1-7, pp. 6093-6096, 2004

[22] J.L. McClelland and A.H. Kawamoto, "Mechanisms of sentence processing: assigning roles to constituents of sentences," D.E. Rumelhart, J.L. McClelland, the PDP Research (Eds.), Parallel Distributed Process 2, MIT Press, pp. 272-325, 1986.

[23] M. McHale, "A Comparison of WordNet and Roget's taxonomy for measuring semantic similarity," in: Proc. COLING/ACL Workshop Usage of WordNet in Natural Language Processing Systems, Montreal, 1998.

[24] C.T. Meadow, B.R. Boyce, and D.H. Kraft, Text Information Retrieval Systems. 2nd Ed, Academic Press, 2000.

[25] D. Michie, "Return of the imitation game," Electronic Transactions on Artificial Intelligence, vol. 6, no. 2, pp.203-221, 2001.

[26] G.A. Miller, "WordNet: A lexical database for English," Communications of the ACM, vol. 38, no. 11, pp. 39-41, 1995.

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

[28] N. Okazaki, Y. Matsuo, N. Matsumura, and M. Ishizuka, "Sentence extraction by spreading activation through sentence similarity," IEICE Transactions on Information and Systems, vol. E86D, no. 9, pp. 1686-1694, 2003.

[29] E.K. Park, D.Y. Ra, and M.G. Jang, "Techniques for improving web retrieval effectiveness," Information Processing & Management, vol. 41, no. 5, pp. 1207-1223, 2005.

[30] R. Rada, H. Mili, E. Bichnell, and M. Blettner, "Development and application of a metric on semantic nets," IEEE Transactions on System, Man, and Cybernetics, vol. 9, no. 1, pp. 17-30, 1989.

[31] A. Radford, M. Atkinson, D. Britain, H. Clahsen, and A. Spencer, Linguistics: An Introduction. Cambridge University Press, 1999.

[32] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy," 14th International Joint Conference on AI, 1995.

[33] F.J. Ribadas, M.Vilares, and J. Vilares, "Semantic similarity between sentences through approximate tree matching," Lecture Notes in Computer Science, vol. 3523, pp. 638-646, 2005.

[34] M.A. Rodriguez and M.J. Egenhofer, "Determining semantic similarity among entity classes from different ontologies," IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 2, pp. 442-456, 2003.

[35] H. Rubenstein and J.B. Goodenough, "Contextual Correlates of Synonymy," Comm. ACM, vol.8, No. 10, pp627-633, 1965.

[36] G. Salton, Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, Mass.Wokingham, 1989.

[37] J. Sinclair (Ed.), Collins Cobuild English Dictionary for Advanced Learners. 3rd. Edition, Harper Collins Pub. ISBN 0-00-375115-5, 2001.

[38] The Gene Ontology Consortium, "Gene Ontology Software and Databases," available at: http://www.geneontology.org/GO.doc.shtml.

[39] USGS, "View the SDTS Document," available at: http://mcmcweb.er.usgs.gov/sdts/standard.html.