International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 24 (2016) pp. 11755-11762
© Research India Publications. http://www.ripublication.com

Text-Fragment Similarity Measurement using Word Sense Identification

Khaled Abdalgader

Faculty of Computing and Information Technology, Sohar University
PO Box 44, PCI 311, Sohar, Sultanate of Oman.

Abstract
While measures of text similarity have been in existence since the inception of information retrieval, a growing number of application areas rely on calculation of similarity between short text fragments. Conventional text similarity measures such as Jaccard, Dice and Cosine similarity, which are based on word co-occurrence, are not generally suitable for this task because two short text fragments may be semantically similar despite having no words in common. Measures of short text similarity must therefore take into account linguistic information contained in the text fragments under comparison. While a number of short text similarity measures have recently been proposed, these methods have not incorporated word sense disambiguation, and the research described in this paper is motivated by the belief that the performance of such measures could be improved through the correct identification of word sense. The paper proposes a novel, computationally efficient word sense disambiguation method that operates by comparing the WordNet glosses of the target word with a context vector comprising the remaining words in the text fragment surrounding the target word. Unlike most word sense disambiguation algorithms, the method is not based on word overlap, but itself uses a measure of short text similarity. The paper then shows how this word sense disambiguation technique can be incorporated within a short text similarity measure, and evaluates the resulting method on the benchmark Microsoft Research Paraphrase Corpus. Results show that the incorporation of word sense disambiguation leads to a significant improvement in performance.

Keywords: WordNet, word sense disambiguation, fragments similarity.

INTRODUCTION
Measuring the similarity between small-sized text fragments (e.g., sentences) is a fundamental function in many textual applications. These include text mining and text summarization, which usually operate at the sentence or sub-sentence level; question answering, where it is necessary to calculate the similarity between a question-answer pair; and image retrieval, where we are interested in the similarity between a query and an image caption.
Although methods for measuring text similarity have been in existence for decades, most approaches are based on word co-occurrence. The assumption here is that the more similar two texts are, the more words they have in common. While this assumption is generally valid for large-size text fragments (e.g., documents)—and hence the widespread and successful use of these methods in information retrieval—the assumption does not hold for small-sized text fragments such as sentences. This is because the flexibility of natural language enables humans to express similar meaning using sentences that may be quite different not only in their structure, but also in regard to their component words. That is, two sentences may be semantically very similar despite having few, if any, words in common. Since our interest is in measuring similarity between short text fragments, we concern ourselves in this paper with linguistic measures of text similarity; i.e., text similarity measures which take into account the semantic information associated with the component words of the text fragments being compared.
Various linguistic measures for short text similarity have been proposed in recent years [1-5], and all of these define the similarity of two text fragments as being some function of the semantic similarities between their constituent words. However, many words have more than one meaning (polysemony), and in order to accurately calculate the similarity between two text fragments, it is therefore important to correctly identify the sense in which the words are being used in those fragments. This problem is known as word sense disambiguation (WSD). The correct sense of a word needs to be determined in the context of the text fragment in which it appears, and this presents yet another difficulty, since short text provides only a very limited context.
While a number of papers have reported on the development and evaluation of short text similarity measures, to our knowledge there has not yet been any attempt to incorporate word sense disambiguation into these measures. The research described in this paper is motivated by the belief that the performance of such measures could be improved through the correct identification of word sense.
The contributions of this paper are two-fold. Firstly, we present a novel knowledge-based word sense disambiguation algorithm that operates by computing the similarity between WordNet [6] glosses of the target polysemous word and a context vector made up of all of the remaining words in the text fragment. The target word is assigned the sense associated with the gloss which has the highest similarity score to the context vector, and the procedure is then repeated for all words in the text-fragment. Secondly, we show how the word sense disambiguation technique can be incorporated within a variant of Li e*t al's* (2006) [2] text similarity measure, and evaluate the resulting method on the benchmark Microsoft Research Paraphrase Corpus [7]. Results show that

the incorporation of word sense disambiguation does lead to a significant improvement in performance.

The remainder of the paper is structured as follows. Section 2 provides relevant background into linguistic text similarity measures and word sense disambiguation methods. Section 3 presents our word sense disambiguation method, and Section 4 describes how this method can be incorporated into a variant of Li *et al's* (2006) text similarity measure. Empirical results are presented in Section 5, and Section 6 contains a discussion and closing remarks.

BACKGROUND

Linguistic Text Similarity Measures

A number of linguistic text similarity measures have recently been proposed in the literature. The approach proposed by Li *et al* (2006)[2] is based on the notion of semantic vectors. Sentences are first transformed into feature vectors having words from the sentence pair as a feature set. That is, rather than using the full set of features from some corpora, only the words appearing in the two sentences are used, thus overcoming the problem of data sparseness arising from the high dimensional vector space of a full bag-of-words representation. Word weights for the semantic-vectors are derived from the maximum semantic similarity score between words in the feature vector and words in the corresponding sentence. If a word from the feature vector appears in the corresponding sentence, then a weight of 1 is assigned for that word; otherwise, word-to-word similarity scores are calculated between the target word (i.e., the word whose weight is being computed) and all of the words from the corresponding sentence, and the weight is determined as the maximum of these similarity scores. Finally, the semantic similarity between the pair of semantic-vectors is defined as a cosine of the angle between the two vectors, as per the traditional vector-space approach. Note that Li *et al*'s approach also utilizes word order in the similarity computation, with the final similarity measure being a linear combination of semantic vector similarity and word order similarity, controlled by a mixing coefficient.

The method proposed in Mihalcea *et at* (2006)[3] combines word semantic similarity scores with word specificity scores. Given two text fragments s_1 and s_2, the similarity computation begins by calculating the similarity score between the first word in s_2 and each word in s_1 that belongs to the same part of speech class. The maximum of these scores is then weighted with the *idf* score of the word from s_2. This procedure is then repeated for the remaining words in s_2, with the weighted maximum scores summed, and then normalized by dividing by the sum of *idf* scores for words in s_1. This procedure is then repeated for s_1. The overall similarity is defined as the average of normalized weighted maximums for s_1 and s_2.

$$sim_{sem,IDF}(s_1,s_2) = \frac{1}{2} f(T_1,T_2) + \frac{1}{2} f(T_2,T_1)$$

$$f(T_a,T_b) = \sum_{w\in\{s_1\}} \left(\max Sim(w,s_2) \times idf(w)\right) \Big/ \sum_{w\in\{s_1\}} idf(w)$$

Note that while Mihalcea *et al*'s method also restricts the set of feature words to those appearing in the two fragments being compared, the *idf* score is determined using an external corpus. Also note that the reason for computing the semantic similarity scores only between words in the same part of speech class is that most WordNet-based measures are unable to calculate semantic similarity of words belonging to different parts of speech.

Ramage *et al* (2009)[5] have very recently introduced a variant of the vector space model based on the idea of random walks over a graph derived from WordNet [6], together with statistical information from a corpus. Instead of comparing vectors for each text fragment directly, their method compares the distribution each fragment induces when used as the seed for a random walk over the graph. Once stationary distributions have been reached, the distributions are compared using standard vector similarity measures such as Cosine similarity or Jaccard score.

Another recent contribution is from Achananuparp *et at* (2009) [1], who propose a novel approach that employs the semantic structure of sentences in the form of verb argument structure to measure the semantic similarity between sentence pairs. Their approach was motivated by the intuition that sentences which express the same meaning should have similar verb argument structure. Thus, instead of comparing two unstructured sentences, their method decomposes sentences into a set of verb argument roles. Given two sentences s_1 and s_2, the similarity score between the verb argument role k of sentence s_1 and verb argument role l of sentence s_2 is estimated by the similarity between their verbs and the sum of similarities between the corresponding arguments.

Each of the measures defined above depends in some way on a measure of semantic similarity between words. A large number of such measures have been proposed, and most of these rely on semantic relations expressed in resources such as dictionaries, thesauri, or lexical knowledge-bases such as WordNet. We describe several WordNet-based word-to-word semantic similarity measures in Section 3.3. For a comprehensive overview, see Budanitsky & Hirst (2001) [8] or Budanitsky & Hirst (2006) [9].

Word Sense Disambiguation

Many words are polysemous, and can take on multiple meanings, depending on the context in which they are used. Word sense disambiguation is the process of assigning the correct meaning to a word, based on the context in which it appears. For example, if the words *deposit*, *money* or *loan* appear near the word *bank*, humans can easily identify that the intended sense of *bank* is the financial institution, and not the slope beside a body of water.

Techniques for Word sense disambiguation generally fall into one of two families: corpus-based methods, and knowledge-based methods. Corpus-based methods utilize supervised learning techniques to induce a classifier from a corpus of training data consisting of a set of labeled sentences in which the label indicates the sense in which the respective word is being used. Once a classifier has been created, it can then be used to predict the sense of the target word in novel sentences. In contrast, knowledge-based methods do not usually require any such corpora, relying instead on external resources such as dictionaries, thesauri, or lexical knowledge-bases such as WordNet [6]. While corpus-based methods have generally been found to perform more accurately than knowledge-based methods, the fact that a separate classifier must be induced for every word severely limits the coverage of these methods. Therefore, for general applications such as text-mining or document summarization, where wide-ranging coverage is required, knowledge-based methods tend to be preferred.

One of the first attempts to automate word sense disambiguation was by Lesk (1988) [10], who employed a dictionary-based (i.e., knowledge-based) approach. The method determines the sense of a polysemous word by calculating the word overlap between the glosses (i.e., definitions) of two or more target words. The correct sense of the target words are assumed to be those whose glosses have the greatest word overlap. For example, in the case of two words w_1 and w_2, the Lesk score is defined as

$$\text{Score}_{Lesk}(S_1, S_2) = |gloss(S_1) \cap gloss(S_2)|,$$

where $S_1 \in Senses(w_1)$, $S_2 \in Senses(w_2)$ and $gloss(S_i)$ is the bag of words in the definition of sense S_i of w_i. The senses which score the highest value from the above calculation are assigned to the respective words.

While the Lesk algorithm is feasible when the context is small (e.g., two words) it leads to combinatorial explosion as the number of words increases. For example, in a two-word context the number of gloss overlap calculations is $|senses(w_1)| \cdot |senses(w_2)|$, whereas in the case of an n-word context, this increases exponentially to $|senses(w_1)| \cdot |senses(w_2)| \cdot \ldots \cdot |senses(w_n)|$. For this reason, a simplified version of this approach is commonly used, in which the correct sense for word w is selected as the one whose gloss has the greatest overlap with the words in the context of w. That is,

$$\text{Score}_{LeskVar}(S) = |context(w) \cap gloss(S)|,$$

where $context(w)$ is the bag of words in a context window that surrounds the word w.

Lesk's algorithm suffers from the fact that dictionary glosses are often quite brief, and may not include sufficient vocabulary to identify appropriate senses. To alleviate this problem, Banerjee and Pedersen (2002) [11] proposed an adaptation of the Lesk method based on the use of WordNet. Rather than simply considering the glosses of the surrounding words in the text fragment, the concept hierarchy of WordNet is exploited to allow for glosses of senses related to the words in the context to be compared as well. In effect, the glosses of surrounding words in the text-fragment are expanded to include glosses of those words to which they are related through relations in WordNet. The range of relationships (e.g., hyperonymy, meronymy, etc.) applied to extend the glosses is a parameter, and can be chosen from any combination of WordNet relations. In applying their measure of word relatedness to word sense disambiguation, Banerjee and Pedersen (2002) [11] use a window of context around the target word, and assign to each of the possible senses a score computed by adding the relatedness scores obtained by comparing the sense of the target word with every sense of the non-target words in the context window. The correct sense of the target word is determined as that with the highest score.

While the Lesk algorithm and many of its variants are based on the notion of gloss overlap, Patwardhan *et al* (2003) [12], following Resnik (1995) [13], take the view that gloss overlaps are just another measure of semantic relatedness, and evaluate the use of several semantic measures in place of gloss overlap. They use the context window approach, as described immediately above in relation to Banerjee and Pedersen (2002) [11], to determine the sense of the target word with the highest score, and find that of the five semantic measures tested, the best performance is obtained using the Jiang and Conrath [14] measure (described in Section 3.3).

The method proposed by Sinha and Mihalcea (2007) [15] operates by associating with each word in a text fragment a list of its possible senses. The senses of nearby words are then linked together using one of six different semantic similarity measures, forming a graph over the text fragment, with vertices being word senses, and edges weighted by the similarity between senses. One of four graph centrality algorithms is then used to determine the importance of vertices in the graph, assigning each vertex a score which is then used to identify the most probable sense for each word. In comparison to the approaches of Banerjee and Pedersen (2002) [11] and Patwardhan *et al* (2003) [12], in which words are disambiguated one at a time using a small window of context surrounding the target word, Sinha and Mihalcea's approach disambiguates all words in the text fragment simultaneously. This means that it is likely to result in a more coherent set of meanings over all of the words in the text fragment. However, the approach is computationally intensive, and requires controlling the complexity by only linking words that appear within some fixed distance from each other in the text fragment.

In the next section we propose an approach that disambiguates words one at a time, but using the context provided by the entire text fragment.

WORD SENSE DISAMBIGUATION METHOD

In line with Lesk algorithm variant described above, the method proposed in this section disambiguates words one at a time; however, rather than using the context provided only in some fixed-size context window surrounding the target word, the method disambiguates the target word using the context provided by all remaining words in the text fragment. Essentially, the algorithm computes the similarity between WordNet glosses of the target polysemous word and the text made up of all of the remaining words in the text fragment, which we refer to as the *context vector*. The target word is then assigned the sense associated with the gloss which has the highest similarity score to the context vector. This procedure is then repeated for all words in the text-fragment.

Note that the disambiguation algorithm itself relies upon a measure of text similarity, since a gloss (which is a text fragment) must be compared with a context vector (another text fragment). The situation is thus somewhat circular, as our motivation for introducing word sense disambiguation was to improve the measurement of short-text similarity. Attempting to disambiguate the sense of polysemous words in the gloss and context vectors would lead to an infinite regress. Thus, we only perform disambiguation at the top level; i.e., only to polysemous words in the original text fragments s_1 and s_2. We now describe the word sense disambiguation algorithm in detail.

The text fragment containing the words to be disambiguated is first represented as the set of non-stopwords that it contains:

$$W = \{w_i \mid i=1..N\}$$

where N is the number of words in W. Stopwords are removed because they do not carry any semantic information. Suppose that we wish to determine the sense for word w_i from W. Let G_{w_i} be the set of WordNet glosses corresponding to word w_i,

$$G_{w_i} = \left\{ g_{w_i}^k \mid k = 1..N_{w_i} \right\}$$

where N_{w_i} is the number of WordNet senses associated with w_i, and $g_{w_i}^k$ is the set of non-stopwords in the k-1th WordNet gloss for w_i (WordNet indexing starts from 0). Let R_i be the context vector comprising all words from W, except w_i :

$$R_i = \left\{ w_j \mid w_j \in W \text{ and } j \neq i \right\}$$

The sense for word w_i is identified as the k value for which $g_{w_i}^k$ is most similar to R_i. The procedure for disambiguating all polysemous words from some text fragment is described in Algorithm 1.

Algorithm 1. Word sense identification

```
Input:   Words $W = \{w_i \mid i = 1..N\}$
         Glosses $G_{w_i} = \{g_{w_i}^k \mid k = 1..N_{w_i}\}, i = 1..N$
Output:  WordNet senses
         $T = \{t_i \mid i = 1..N\}$ where $t_i$ is the WordNet sense of $w_i$
Word Sense Identification
1: for $i$ = 1 to $N$ do
2:     $R_i = \{w_j \mid w_j \in W \text{ and } j \neq i\}$
3:     max_sim ← 0
4:     $t_i \leftarrow 1$
5:     for $j$ = 1 to $N_{w_i}$ do
6:         tmp ← similarity(morphy($g_{w_i}^j$), morphy($R$))
7:         if tmp > max_sim then
8:             max_sim ← tmp
9:             $t_i \leftarrow j$
10:        end if
11:    end for
12: end for
```

Note that in line 6, in which the similarity is calculated, that a function 'morphy' has been applied to both the gloss and context vectors before the similarity is calculated. This function takes a set of words as input, and returns a set of the same length consisting of the morphological stems of the original words. The issue of morphological reduction is important, and at this point we simply highlight the fact that similarity measurement here takes place in the context of determining the sense of a polysemous word. Since we do not wish to disambiguate polysemous words in the gloss and context vectors, instead relying on comparison based on the first WordNet sense (discussed in next sub-section), we similarly opt for the simpler approach of using morphological stems for words in these vectors. In contrast, we will see in Section 4 that in the determining the similarity between the original text fragments s_1 and s_2, that it is indeed important that no such morphological reduction is applied to the original words.

Similarity between Gloss and Context Vectors

Linguistic Text Similarity Measure's Section described several recently-proposed linguistic text similarity measures. In performing word sense disambiguation, as described in Algorithm 1, we use a variation on the method proposed by Li *et al* (2006) [2] to calculate the similarity measure in line 6.

Let W_1 and W_2 be the word sets of the two text fragments whose similarity we wish to calculate. Assume that W_1 is the gloss vector corresponding to the WordNet sense k for the target word w_i and W_2 is the corresponding context vector:

$$W_1 = g_{w_i}^k = \left\{ w_{1i} \mid i = 1..N_1 \right\}$$

$$W_2 = R_i = \left\{ w_{2i} \mid i = 1..N_2 \right\}$$

We first form the union word set U by combining all distinct words from W_1 and W_2:

$$U = W_1 \cup W_2 = \left\{ w_i \mid i = 1..N \leq N_1 + N_2 \right\}$$

where N is the total number of words in U. We now construct semantic vectors $\mathbf{V}_1$ and $\mathbf{V}_2$, corresponding to W_1 and W_2 respectively. Each entry of these vectors corresponds to a word in the union set U, so their dimension is N. Let v_{ij} be the jth element of $\mathbf{V}_i$, and let w_j be the corresponding word from U. The value of v_{ij} is determined according to the semantic similarity of w_j to all words in W_i. There are two cases to consider, depending on whether w_j appears or does not appear in W_i :

Case 1: w_j appears in W_i.
Set v_{ij} equal to 1.

Case 2: w_j does not appear in W_i.
Calculate a semantic similarity score between w_j and each word in W_i. Assign the highest of these scores to v_{ij}.

The semantic similarity score used in Case 2 is the 'shortest path similarity', defined in Section 3.3. Once $\mathbf{V}_1$ and $\mathbf{V}_2$ have been determined, the semantic similarity is measured as the Cosine similarity between $\mathbf{V}_1$ and $\mathbf{V}_2$:

$$\text{similarity}(W_1, W_2) = (\mathbf{V}_1 \cdot \mathbf{V}_2) / (|\mathbf{V}_1||\mathbf{V}_2|)$$

For clarity, we now provide an example of this procedure.

AWalk-Through Example

Consider the sentence "*The virus infected all saving deposit money systems in the bank*" and suppose that we wish to disambiguate the word 'virus'. We first construct the set of non-stopwords appearing in the sentence:

W = {'virus', 'infected', 'saving', 'deposit', 'money', 'systems', 'bank'}

The word 'virus' has three WordNet glosses:

Sense 1: (virology) ultramicroscopic infectious agent that replicates itself only within cells of living hosts; many are pathogenic; a piece of nucleic acid (DNA or RNA) wrapped in a thin coat of protein

Sense 2: a harmful or corrupting agency

Sense 3: a software program capable of reproducing itself and usually capable of causing great harm to files or other programs on the same computer

We construct the sets g^1_{virus}, g^2_{virus} and g^3_{virus} corresponding to each of these glosses. For space reasons, we show only the second:

g^2_{virus} = {'harmful', 'corrupt', 'agency'}

Since we are disambiguating the word 'virus', the context vector R_{virus} will consist of all words from W except 'virus':

R_{virus} = {'infect', 'saving', 'deposit', 'money', 'system', 'bank'}

The union set is formed by combining words from g^1_{virus} and R_{virus}:

U = {'saving', 'infect', 'agency', 'system', 'harmful', 'deposit', 'money', 'corrupt', 'bank'}.

Note that in constructing the union set, morphological stemming has been applied to the words from W_1 and W_2, as per line 6 of Algorithm 1, but has not been applied to the target polysemous word.

We now calculate the semantic vectors $\mathbf{V}_1$ and $\mathbf{V}_2$, corresponding to g^2_{virus} and R_{virus} respectively. As an example, consider the word *saving*, which is the first word in the union set. The word 'saving' does not appear in g^2_{virus}, therefore we calculate a semantic similarity score between 'saving' and each of the words in g^2_{virus}. (Recall that we use the first WordNet sense for each of these words). The maximum similarity score is 0.083, so this becomes the value corresponding to 'saving' in $\mathbf{V}_1$. In contrast, 'saving' does appear in R_{virus}, so the value corresponding to 'saving' in $\mathbf{V}_2$ becomes 1.0. Repeating this procedure for remaining words results in

V_1 = (0.083, 0.0, 1.0, 0.071, 1.0, 0.071, 0.083, 1.0, 0.076)
V_2 = (1.0, 1.0, 0.083, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0)

Finally, we calculate the cosine of these vectors, resulting in a similarity score of 0.110. Using the first and third glosses results in similarity scores of 0.191 and 0.224, and thus the sense of word *virus* in the original sentence will be determined as that corresponding to the third WordNet sense, which is clearly the sense in which *virus* would be interpreted by a human in this context.

Repeating this same procedure for all words in W results in the following set of sense-assigned words:

{('virus', 2), ('infected', 4), ('saving', 7), ('deposit', 3), ('money', 2), ('systems', 0), ('bank', 1)}

where the numbers indicate the sense of the associated word. The senses are as follows:

virus, 2 (n):	*a software program capable of reproducing itself and usually capable of causing great harm to files or other programs on the same computer*
infected, 4 (a):	*containing or resulting from disease-causing organisms*
saving, 7 (v):	*accumulate money for future use*
deposit, 3 (n):	*money deposited in a bank or some similar institution*
money, 2 (n):	*the official currency issued by a government or national bank*
systems, 0 (n):	*instrumentality that combines interrelated interacting artifacts designed to work as a coherent entity*
bank, 1 (n):	*a financial institution that accepts deposits and channels the money into lending activities*

where the characters in parentheses indicate the part of speech (i.e., noun, adjective, verb, etc.). With the exception of the sense assigned to the word '*saving*', all of these assignments are in accord with the meaning assigned by a human.

Word-to-Word Semantic Similarity

The algorithm described above relies on a semantic word similarity measure. A number of such measures have been defined. In the following we describe the four measures we use in this paper, all of which are based on WordNet.

The shortest path similarity [4] is the simplest measure we use, and is defined as:

$$Sim_{Path}(w_1, w_2) = \frac{1}{length(w_1, w_2)}$$

where *length* is the length of the shortest path between two words using node-counting (including the end nodes) in the WordNet hierarchy. Of the four measures described in this section, only the shortest path measure is capable of calculating the similarity between words with different part of speech; e.g., a noun and an adjective. This reason, together with the computational simplicity of the approach, makes it appropriate to use within Algorithm 1.

A problem with the shortest path measure is that it does not take into account the depth of words in the hierarchy. Since words at the top levels of the hierarchy have more general semantics and less similarity between them than words at lower levels, the depth of words in the hierarchy should be taken into account. The following three methods do take into account this depth.

The measure proposed by Resnik (1995) [13] is based on the idea that the degree to which two words are similar is proportional to the amount of information they share. The measure is defined as the information content of the Lowest Common Subsumer (LCS) of the two words:

$$Sim_{res}(w_1, w_2) = IC(LCS(w_1, w_2))$$

The LCS of two words is the words' deepest common ancestor in the hierarchy. The information content, $IC(w)$, is a measure of the specificity of a word, and is defined as $IC(w) = -\log P(w)$, where $P(w)$ is the probability that word w appears in a large corpus.

The Lin (1998) [16] measure normalizes the Resnik measure by dividing it by the average information content of the individual words w_1 and w_2:

$$Sim_{Lin}(w_1, w_2) = \frac{2 \times IC(LCS(w_1, w_2))}{IC(w_1) + IC(w_2)}$$

The measure proposed by Jiang and Conrath (J&C) (1997) [14] is also based on information content and is defined as:

$$Sim_{Jcn}(w_1, w_2) = \frac{1}{IC(w_1) + IC(w_2) - 2 \times IC(LCS(w_1, w_2))}$$

Note that all of the above measures are normalized to fall within a 0 to 1 range.

TEXT-FRAGMENT SIMILARITY MEASURE

We now show how the word sense disambiguation method described above can be incorporated within a text fragment similarity measure.

Let s_1 and s_2 be the two text fragments whose similarity we wish to measure. We first represent each of these text fragments as the set of non-stopwords that they contain; i.e.,

$$W_1 = \{w_{1i} \mid i=1..N_1\}$$
$$W_2 = \{w_{2i} \mid i=1..N_2\}$$

where N_1 and N_2 are the number of words in W_1 and W_2 respectively. Note that we do not perform morphological stemming on the words from s_1 and s_2. The main reason for this is that we wish to preserve as much of the meaning of the original words as possible, and this means that we should keep words in their original form (e.g., noun, verb, etc.).

We then use the method described in Section 3 to disambiguate all polysemous words in W_1 and W_2. This results in two sets of sense-assigned words:

$$S_1 = \{w_{1i}^{t_{1i}} \mid i=1..N_1\}$$
$$S_2 = \{w_{2i}^{t_i} \mid i=1..N_2\}$$

where w_{ji} is the ith word from W_1, and t_{ji} indicates the identified WordNet sense for this word.

Once again, we form the union set U, and use it to construct the semantic vectors $\mathbf{V}_1$ and $\mathbf{V}_2$. However, in constructing the union set, we note that it may be possible for the same word to appear more than once. This is because the input sets are sense-assigned, and it is thus possible for the same word to appear more than once, in each case with a different WordNet [6] sense. This has implications for determining the semantic vector values. Let v_{ij} be the jth element of $\mathbf{V}_i$, and let $w_j^{t_j}$ be the corresponding sense-assigned word from U. The two cases to consider depend on whether or not the sense-assigned word appears in S_i :

Case 1: $w_j^{t_j}$ appears in S_i.

The word appears with the same sense in S_i, therefore we set v_{ij} equal to 1.

Case 2: $w_j^{t_j}$ does not appear in S_i.

The word w_j either does not appear in S_i, or else it appears in S_i, but with a different sense. Calculate a semantic similarity score between $w_j^{t_j}$ and each word in S_i that has the same part of speech as $w_j^{t_j}$. Assign the highest of these score to v_{ij}.. The complete procedure is described in Algorithm 2.

Algorithm 2. Text fragment similarity

Input: Two sets of sense assigned words, $S_1 = \{w_{1i}^{t_{1i}} \mid i=1..N_1\}$, $S_2 = \{w_{2i}^{t_i} \mid i=1..N_2\}$

Output: Similarity between S_1 and S_2

Text-fragment Similarity Calculation

```
1: $U = S_1 \cup S_2 = \{w_i^{t_i} \mid i=1..N \le N_1 + N_2\}$
2: $\mathbf{V}_1 = (v_{11}, v_{12}, ..., v_{1N})$  // semantic vector for text fragment 1
3: $\mathbf{V}_2 = (v_{21}, v_{22}, ..., v_{2N})$  // semantic vector for text fragment 2
4: for $i$ = 1 to $N$ do
5:     if $w_i^{t_i} \in S_1$ then
6:         $v_{1i}$ = 1
7:     else
8:         sim ← 0
9:         for $j$ = 1 to $N_1$ do
10:            if part_of_speech( $w_i^{t_i}$ ) = part_of_speech ( $w_{1j}^{t_{1j}}$ )
11:                tmp = similarity( $w_i^{t_i}$ , $w_{1j}^{t_{1j}}$ )
12:                if tmp > sim
13:                    sim ← tmp
14:                end if
15:            end if
16:        end for
17:        $v_{1i}$ ← sim
18:    end if
19:    if $w_i^{t_i} \in S_2$ then
20:        $v_{2i}$ = 1
21:    else
22:        sim ← 0
23:        for $j$ = 1 to $N_2$ do
24:            if part_of_speech ( $w_i^{t_i}$ ) = part_of_speech ( $w_{2j}^{t_{2j}}$ )
25:                tmp = similarity ( $w_i^{t_i}$ , $w_{2j}^{t_{2j}}$ )
26:                if tmp > sim
27:                    sim ← tmp
28:                end if
29:            end if
30:        end for
31:        $v_{2i}$ ← sim
32:    end if
33: end for
34: similarity $(s_1, s_2) = (\mathbf{V}_1 \cdot \mathbf{V}_2) / (|\mathbf{V}_1||\mathbf{V}_2|)$
```

EXPERIMENTAL RESULTS

We have applied the method described above to the Microsoft Research Paraphrase Corpus (MSRP)[7], which has become a benchmark dataset used in evaluating short text similarity measures. This corpus consists of 5801 pairs of text-fragments that were automatically collected from a large number of newswire postings on the web over a period of 18 months. Each pair of text fragments was manually labeled by two human annotators with a binary true or false value, indicating whether or not the two fragments in a pair were considered a paraphrase of each other. The agreement between the human judges was 83%. The corpus is divided into 4076 training pairs and 1725 test pairs. Note that since this is an unsupervised experiment, we used only the test data. All results reported are based on the use of the lexical knowledge base WordNet 3.1, which has 150,000 words organized in over 115,000 synsets for a total of 207,000 word-sense pairs.

We use four evaluation measures to evaluate our results: *accuracy*, *precision*, *recall* and *F-measure*. Accuracy is simply the number of correct predictions divided by the number of text fragment pairs. Precision is the number of correct true predictions divided by the total number of true predictions. Recall is the number of correct true predictions divided by the actual number of paraphrase pairs. F-measure is the uniform harmonic mean of precision and recall.

Table 1 shows the performance of the proposed method using the four word-to-word semantic similarity measures described in Section 3.3. The first section of the table shows performance with the use of word sense disambiguation; the second shows performance without word sense disambiguation (i.e., word-to-word semantic similarity is measured based on the first WordNet sense of the component words).

TABLE 1. Performance of Similarity Measure on MSRP Data

Measure	Acc.	Prec.	Rec.	F
Similarity Measure *with* WSD				
Path	69.2	76.7	76.9	76.8
Resnik	66.7	66.9	98.5	79.7
Lin	**72.4**	76.4	84.5	80.3
J&C	69.1	77.2	75.8	76.5
Similarity Measure *without* WSD				
Path	68.6	68.6	97.2	80.5
Resnik	66.4	66.4	1.0	79.8
Lin	67.7	67.6	98.7	80.2
J&C	69.1	69.1	96.5	80.6

When word sense disambiguation is used, the best performance, in terms of both overall accuracy and F-measure (Accuracy = 72.4%, F-measure = 80.3) is achieved using the Lin measure.

In order to compare performance with that of other approaches, the results in Table 1 are based on a 0.5 classification threshold. To determine whether better performance can be obtained using some other threshold, we applied the algorithm using the Lin similarity measure, with and without the use of word sense disambiguation, using 11 threshold values ranging from 0.0 to 1.0. The results are shown in Table 2. Note that the best performance without the use of word sense disambiguation corresponds to a threshold of 0.7. Note also, that by raising the threshold in this case from 0.5 to 0.7, that the precision has increased from 67.6% to 75.0%. This suggests that, on average, the value of the cosine similarity measure between text fragments is higher when word sense disambiguation is not used than when it is used. This can be explained by the fact when using word sense disambiguation, polysemous words may be assigned different senses in each sentence of a pair, thus resulting in a smaller similarity value than would be the case if the words were assigned the same sense.

TABLE 2. Performance Corresponding to Optimal Thresholds for Lin Measure

Threshold	Acc.	Prec.	Rec.	F
With WSD				
0.50	72.4	76.4	84.5	80.3
Without WSD				
0.70	70.7	75.0	84.0	79.2

For comparative purposes, Table 3 shows the performance of a number of other algorithms that have been reported in the literature: Mihalcea *et al*'s (2006) [3] corpus-based and WordNet-based measures; the random graph walk method of Ramage *et al* (2009) [5] using three distributional similarity measures; and, following Mihalcea *et al* (2006) [3], a baseline that measures the cosine similarity between vectors in a full bag-of-words representation with *tf-idf* weighting, and a random baseline, created by randomly assigning a true or false value to pairs of text fragments. Note that none of these methods incorporate word sense disambiguation.

The accuracy of 72.4% achieved by our method on this dataset is higher than that of any of the methods described immediately above, and, to our knowledge, better than any results reported in the literature. As in Mihalcea *et al* (2006) [3], we also calculate the error reduction with respect to the vector space cosine similarity baseline, resulting in an error reduction of 20%, as compared with a value of 13.8% reported by Mihalcea *et al*.

TABLE 3. Performance of other Approaches

Measure	Acc.	Prec.	Rec.	F
Mihalcea *et al.*, Corpus-based				
PMI-IR	69.9	70.2	95.2	81.0
LSA	68.4	69.7	95.2	80.5
Mihalcea *et al.*, WordNet-based				
Lin	69.3	71.6	88.7	79.2
J&C	69.3	72.2	87.1	79.0
Resnik	69.0	69.0	96.4	80.4
Ramage *et al.*, Random Graph Walk				
Cosine	68.7	-	-	78.7
Dice	70.8	-	-	80.1
JS	68.8	-	-	80.5
Baselines				
Vector-based	65.4	71.6	79.5	75.3
Random	51.3	68.3	50.0	57.8

DISCUSSION AND CONCLUSIONS

One of the advantages of the word sense disambiguation algorithm that we have presented is that it utilizes the context provided by all words in the text-fragment, and not just those within some fixed-size window surrounding the target word. Importantly, this does not lead to significantly increased computational requirements. To demonstrate, it is useful to compare the computational complexity of our approach with that of the context window approach used in Banerjee and Pedersen (2002) [11] and Patwardhan *et al* (2003) [12]. Suppose that the average number of WordNet senses per word is a, the average length of text fragments and WordNet glosses (after removal of stopwords) is N, and in the case of a context-window approach, that the number of words used on either size of the target word is n. Since both approaches disambiguate words one at a time, we consider the cost of disambiguating a single word. The number of word-to-word similarity calculations required to disambiguate a word using the context window approach is quadratic in a (since all senses of the target word must be compared with all sense of the neighboring words), and linear in n. In our approach, the number of similarity calculations is linear in a (since only the various senses of the target word are considered, with first sense used for words in the context vector), but is quadratic in n (since determining the semantic vectors requires calculating the similarity between words in the union set with words appearing in the original sentences). However, since the fragments are short, and since stopwords have been removed, the values of n and a will be comparable, and thus the computational requirements of the proposed approach will usually not be much greater than those of the context window approach. If computation requirements are prohibitive, it is of course possible to also adopt a windowing approach within our method, use the context provided by all words within this window.

Our approach does not take into account the various possible senses for words in the context vector, and uses only the first WordNet sense for these words in disambiguating some target word [17]. While this may appear risky, in that some other sense of words in the context vector may have resulted in a more accurate disambiguation of the target word, the fact that all words in the context vector are being used reduces this risk. In fact, given that all words in the text fragment are to be disambiguated, there is no reason why a second pass of the algorithm cannot be applied, in which case the sense-assigned words resulting from the first pass are used in the second pass of the disambiguation algorithm.

Our primary interest in this area lies in the context of sentence clustering. Clustering methods rely on a similarity measure, and hence our focus in this paper on short-text similarity measures. We have demonstrated that the incorporation of word sense disambiguation leads to an improved similarity measure whose performance on the MRSP exceeds that of existing similarity measures. We are currently comparing the performance of the disambiguation algorithm with that of other disambiguation algorithms directly on word sense disambiguation datasets.

REFERENCES

[1] Achananuparp, P., Hu, X., and Yang, C. 2009. Addressing the Variability of Natural Language Expression in Sentence Similarity with Semantic Structure of the Sentences. In *Proc. PAKDD* 2009. Bangkok. 548-555.

[2] Li, Y., McLean, D., Bandar, Z.A., O'Shea, J.D., Crockett, K. 2006. Sentence Similarity Based on Semantic Nets and Corpus Statistics. *IEEE Transactions on Knowledge and Data Engineering* 18(8), 1138–1150.

[3] Mihalcea, R., Corley, C., Strapparava, C. 2006. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence,* Boston. 1 (16–20 July 2006), 775-780.

[4] Abdalgader, K. and Skabar, A. 2011. Short-text similarity measurement using word sense disambiguation and synonym expansion. In *Proceedings of the 23rd Australasian Joint Conference on Artificial Intelligence*. (AI2010, Adelaide). Advances in Artificial Intelligence. 6464, 435-444.

[5] Abdalgader, K. 2014. Word Sense Identification Improves the Measurement of Short-Text Similarity. In *Proceedings of the International Conference on Computing Technology and Information Management* (ICCTIM2014), Dubai, UAE, Digital Library of SDIWC, ISBN: 978-0-9891305-5-4, pages 233-243.

[6] Fellbaum, C. 1998. WordNet: An Electronic Lexical Database. MIT Press, Cambridge.

[7] 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*. 350-356.

[8] Budanitsky, A., and Hirst, G. 2001. Semantic distance in Word-Net: An experimental, application-oriented evaluation of five measures. In *Proceedings of the NAACL Workshop on WordNet and Other Lexical Resources*. 29-34.

[9] Budanitsky, A., and Hirst, G. 2006. Evaluating Wordnet-based measures of lexical semantic relatedness. *Computational* Linguistics. 32, 1. 13-47.

[10] 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 1986*. Toronto, Canada. 24-26.

[11] Banerjee, S. and Pedersen, T. 2002. An adapted Lesk algorithm for word sense disambiguation using Word-Net. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics*. Mexico City. 136-145.

[12] Patwardhan, S., Banerjee, S. and Pedersen, T. 2003. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*. Mexico City. (February), 241-257.

[13] Resnik, P. 1995. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Montreal. 1 (20-25 August 1995), 448-453.

[14] Jiang, J., and Conrath, D. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the 10th International Conference on Research in Computational Linguistics (ROCLING X)*. Taipei, Taiwan. 19-33.

[15] Sinha, R. and Mihalcea, R. 2007. Unsupervised Graph-based word sense disambiguation using measures of word semantic similarity. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2007)*, Irvine, CA. 363-369.

[16] Lin, D. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*. Madison, Wisc. (24-27 July 1998), 296-304.

[17] Abdalgader, K., and Skabar, A. 2012. Unsupervised Similarity-Based Word Sense Disambiguation using Context Vectors and Sentential Word Importance. *ACM Transactions on Speech and Language Processing (TSLP)*. vo. 9, no 2.