A Method for Measuring Sentence Similarity and its Application to Conversational Agents

Yuhua Li, Zuhair Bandar, David McLean and James O'Shea

Intelligent Systems Group, Department of Computing and Mathematics,
Manchester Metropolitan University, Manchester, M1 5GD, England
{y.li, z.bandar, d.mclean, j.d.oshea}@mmu.ac.uk

Abstract

This paper presents a novel algorithm for computing similarity between very short texts of sentence length. It will introduce a method that takes account of not only semantic information but also word order information implied in the sentences. Firstly, semantic similarity between two sentences is derived from information from a structured lexical database and from corpus statistics. Secondly, word order similarity is computed from the position of word appearance in the sentence. Finally, sentence similarity is computed as a combination of semantic similarity and word order similarity. The proposed algorithm is applied to a real world domain of conversational agents. Experimental results demonstrated that the proposed algorithm reduces the scripter's effort to devise rule base for conversational agent.

1 Introduction

Recent applications of natural language processing have a requirement for an effective method to compute similarity between very short texts (such as sentences). Traditionally, techniques for detecting similarity between long texts (documents) have centred on analysing shared words (Meadow, Boyce and Craft 2000). Such methods are usually sufficient for dealing with long texts, because they contain adequate co-occurring words that are sufficient for indicating the text similarity. However, for short texts, word co-occurrence may be rare or not present. This is because people tend to use different sentences to express very similar meanings (Bates 1986). Since such surface information in short texts is very limited, this problem poses a tough challenge for computational methods. The focus of this paper is on computing similarity between very short texts, primarily of sentence length.

Sentence similarity has many interesting applications. Examples include conversational agent with script strategies (Allen 1995; Jurafsky and Martin, 2000) and the Internet.

Although much research has been done on measuring long text similarity, the computation of sentence similarity is far from perfect(Burgess et al. 1998; Foltz et al. 1998; Hatzivassiloglou et al.1999, Landauer et al. 1997). Because of the growing demand from applications, this study is concerned with the development of a method by investigating the underlying information that contributes to the meaning of a sentence. We compute sentence similarity using semantic knowledge from a lexical database and statistical information from a corpus. The impact of syntactic information is also considered in the calculation of similarity. The proposed algorithm differs from existing methods in two aspects. Firstly, we strictly consider text in sentence units, so the surface information is very limited compared to that in document units. Secondly we investigate a method to incorporate word order information in the detection of syntactic similarity.

The next section presents the proposed method for measuring sentence similarity. Section 3 carries out experiments with a conversational agent. Section 4 concludes that the proposed method provides an efficient technique for knowledge representation and management.

2 The Proposed Text Similarity Method

The proposed method derives text similarity from semantic information and syntactic information that are 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 give the text its specific meaning. Texts considered in this paper are assumed to be very short (of sentence length). The task is to establish a computational method that is able to measure the similarity between very short texts (sentences).

2.1 Semantic Similarity between Sentences

Sentences are made up of words, so it is reasonable to represent a sentence using the words in the sentence. The most relevant research area, to our task, is information retrieval. Classical information retrieval methods use a set of a pre-determined index terms (words or collocations) that are used to represent a document in the form of document-term vector. Vector similarity is then used to identify documents that are most related to the query.

Because the index terms are pre-determined and in large numbers, this kind of strategy is inappropriate for computing sentence similarity. A sentence represented using such a large number of pre-determined terms will result in a very sparse vector, i.e., the vector has a very small number of non-zero elements. On the other hand, some important words in a sentence may be missed because of the limits of term set. Unlike classical methods our method dynamically forms the representing semantic vectors solely based on the compared sentences. Recent research achievements in semantic analysis are also adapted to accomplish an efficient semantic vector for a sentence.

Given two sentences:

$$T_1 = \{w_{11} \quad w_{12} \quad \cdots \quad w_{1m_1}\},\ T_2 = \{w_{21} \quad w_{22} \quad \cdots \quad w_{2m_2}\}$$

where w_{ij} is the jth word of T_i (i=1, 2), m_i is the number of words in T_i. A joint-word set $T = T_1 \cup T_2$ is then formed from distinct words in T_1 and T_2:

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

that has m distinct words. It is obvious that $m \leq m_1 + m_2$, because there may be repeated words in a text or between texts.

The joint word set T contains all distinct words in T_1 and T_2. Since inflectional morphology may make a word appear in a sentence with different form that conveys specific meaning for the 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 storage.

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

Since the joint word set is purely derived from the compared sentences, it is compact with no redundant information. This is similar to the index term set in classical information retrieval methods (Meadow, Boyce and Kraft 2000). The joint word set can be viewed as the semantic information for the compared sentences. Overall, the proposed method is derived based on the joint word set of T_1 and T_2. Each sentence is readily represented by the use of a joint word set as follows. The vector derived from the joint word set is called the lexical semantic vector, denoted by $\check{s}$. Each element in this vector corresponds to a word in the joint word set, so its dimension equals the number of words in the joint word set. The value of an element of the 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 example:

Case 1: If w_i appears in T_1, then $\check{s}_i$ 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 (Li, Bandar and McLean 2003), see section 2.2. 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 thresholds is two-fold. Firstly since we use the word similarity of unmatched words, the semantic vector may become noisy. This introduces unwanted information to $\check{s}$ if the maximum similarity is very small. Secondly classical word match methods can be unified into the proposed method by simply setting the threshold to 1. Unlike classical methods, we also keep all function words (Meadow, Boyce and Kraft 2000). This is because function words carry syntactic information that cannot be ignored if a text is very short, e.g. of sentence length. Although function words are retained in a joint word set, they contribute less to the meaning of a sentence than other words. Furthermore each of the words contributes differently to the meaning of the sentence. Thus a scheme is needed to weight each word. We weight the significance of a word using its information content (Meadow, Boyce and Kraft 2000).

It is commonly accepted that frequently used words are less informative than sparsely used ones. The information content of a word is derived from its probability in a corpus (Li, Bandar and McLean 2003). Each cell is weighted by the associated information $I(w_i)$ and $I(\tilde{w}_i)$. Finally the value of an element of the semantic vector is:

$$s_i = \tilde{s} \cdot I(w_i) \cdot I(\tilde{w}_i) \quad (1)$$

where w_i is a word in the joint word set, $\tilde{w}_i$ is its associated word in the sentence. The semantic similarity between two sentences is defined as a cosine coefficient between the two vectors:

$$S_s = \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{\|\mathbf{s}_1\| \cdot \|\mathbf{s}_2\|} \quad (2)$$

2.2 Word Similarity

Given two words: w_1 and w_2, we need to find the semantic similarity of $s_w(w_1, w_2)$ for these two words. In WordNet (Miller 1995) words are organised into synonym sets (synsets), with semantic and relation pointers to other synsets (arcs). We can find the first class in the hierarchical semantic network that subsumes the two synsets for the compared words and take the distance in arcs traversed. 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 and so the depth of word in the hierarchy is taken into account. A formula for word similarity was proposed (Li, Bandar and McLean 2003):

$$s_w(w_1, w_2) = e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \quad (1)$$

where l is the shortest path length between w_1 and w_2, h is the depth of subsumer in the hierarchy, $\alpha \in [0,1], \beta \in (0,1]$ are parameters scaling the contribution of shortest path length and depth, respectively. The optimal parameters for the proposed measure were found to be: α=0.2, β=0.45.

2.3 Word Order Similarity between Sentences

Let's consider a particular case to illustrate the importance of word order. For example, for two sentences:

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

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

These two sentences contain exactly the same words and most words appear in the same order. The only difference is that *dog* appears before *fox* in T_1 and *dog* appears after *fox* in T_2. Since these two sentences contain the same words, any methods based on "bag of word" 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 difference in word order. Therefore any efficient computational method for sentence similarity must take into account the impact of word order.

Sentences containing the same words but in different orders may result in very different meanings. It is easy for humans to process word order information. However the incorporation of order information in to computational methods for understanding natural language is a difficult challenge. This may be the reason why most existing methods do not tackle this type of information. In this section we introduce a method that takes word order information into account when computing sentence similarity.

Assume that for a pair of sentences, the joint word set is T. Recall the above two example sentences, their joint word set is:

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

For each word in T_1 and T_2, a unique index number has been assigned respectively. 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 word order similarity, a word order vector r is formed for T_1 and T_2 respectively based on the joint word set T. For each word w_i in T, we try to find the same or a similar word in T_1 as follows:

1. If T_1 contains an occurrence of the same word, we fill the entry for this word in r_1 with the corresponding index number in T_1. Otherwise we try to find the most similar word $\tilde{w}_i$ in T_1.
2. If the similarity between w_i and $\tilde{w}_i$ is greater than a pre-set threshold, the entry of w_i in 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 r_1 is null.

Having applied the above procedure for T_1 and T_2, the word order vectors for are r_1 and r_2 respectively. For the example sentence pair, we have:

r_1 = {1 2 3 4 5 6 7 8 9},

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:

$$S_r = 1 - \frac{\|\mathbf{r}_1 - \mathbf{r}_2\|}{\|\mathbf{r}_1 + \mathbf{r}_2\|} \quad (3)$$

That is, word order similarity is determined by the normalised difference of word order. The following analysis will demonstrate S_r as an efficient metric for indicating word order similarity. To simplify the analysis, only one word order difference is considered in the following example.

Given two sentences: T_1 and T_2, both sentences have exactly the same words. The only difference between T_1 and T_2 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:

$$a_i = b_i = i \quad \text{for } i=1, 2, \ldots, m \text{ except } i \neq j, j+k$$

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

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

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

$$\|\mathbf{r}_1 - \mathbf{r}_2\|^2 = 2k^2$$

$$\|\mathbf{r}_1 + \mathbf{r}_2\|^2 = 2^2\|r\|^2 + (2.2j.k + k^2) + (-2.2(j+k).k + k^2)$$

then:

$$S_r = 1 - \frac{k}{\sqrt{2\|\mathbf{r}\|^2 - k^2}} \quad (4)$$

We can also derive exactly the same formula for a sentence pair with only one different word, the kth entry. For more general cases with further differences in words and word order, the analytical form of the proposed metric becomes more complicated and we do not intend to present it in this paper. The following features of the proposed word order metric can be revealed:

1. S_r can represent the words shared by two sentences.
2. S_r can represent the order of a pair of 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 increases.
4. For the same number of different words or the same number of word pairs in different orders, S_r is proportional to the sentence length (number of words), its value increases as the sentence length increases. This coincides with intuitive knowledge, i.e. given a fixed number of different words shared between two sentences, the number of identical words shared increases with the length of the sentence.

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

2.4 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 which other words. Both of these semantic and syntactic (in terms of word order) pieces of information play a role in comprehending the meaning of sentences. Thus the overall sentence similarity is defined as a combination of semantic similarity and word order similarity as:

$$\begin{aligned} S(T_1,T_2) &= \delta S_s + (1-\delta) S_r \qquad (5) \\ &= \delta \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{\|\mathbf{s}_1\| \cdot \|\mathbf{s}_2\|} + (1-\delta) \frac{\|\mathbf{r}_1 - \mathbf{r}_2\|}{\|\mathbf{r}_1 + \mathbf{r}_2\|} \end{aligned}$$

where $\delta \le 1$ decides how much semantic and how much word order information contribute to the overall similarity computation. Since syntax plays a subordinate role for semantic processing of text (Wiemer-Hastings 2000), δ should be with a value greater than 0.5, i.e., $\delta \in (0.5,1]$.

Apart from the factor δ, there are two other parameters to set before the similarity can be calculated. These are, a threshold for deriving the semantic vector and a threshold for forming the word order vector. All parameters are empirically set in this paper, 0.4 for word order threshold, 0.2 for semantic threshold and 0.85 for δ.

In the following experiments, we derive semantic information using WordNet (Miller 1995) version 1.6 and word statistical information from the British National Corpus (British National Corpus home page).

3 Applying sentence similarity to Conversational agents

The original motivation for the development of our sentence similarity measure came from the design of a more efficient conversational agent. Thus this section presents experiment results relating to the knowledge management for a conversational agent.

A conversational agent is a system for carrying out a dialogue (usually in natural language) between a human user and a computer agent. The agent usually associates a knowledge base that contains a bank of rules. Computer responses to user’s utterances are governed by matching each user utterance with pattern-based rules embedded in the system. A working rule normally consists of a rule ID, a set of stimulus patterns, the rule's current status and a response-pattern.

1.	*child* bake *cake*
2.	*child* baked *cake*
3.	*child* baking *cake*
4.	*child* cake bake*
5.	*child* cake-bake*
6.	*kid* bake *cake*
7.	*kid* baked *cake*
8.	*kid* baking *cake*
9.	*kid* cake bake*
10.	*kid* cake-bake*
11.	*boys* bake *cake*
12.	*boys* baked *cake*
13.	*boys* baking *cake*
14.	*boys* cake bake*
15.	*boys* cake-bake*
16.	*boy’s* bake *cake*
17.	*boy’s* baked *cake*
18.	*boy’s* baking *cake*
19.	*boy’s* cake bake*
20.	*boy’s* cake-bake*
21.	*boy * bake *cake*
22.	*boy * baked *cake*
23.	*boy * baking *cake*
24.	*boy * cake bake*
25.	*boy * cake-bake*
26.	*girls* bake *cake*
27.	*girls* baked *cake*
28.	*girls* baking *cake*
29.	*girls* cake bake*
30.	*girls* cake-bake*
31.	*girl’s* bake *cake*
32.	*girl’s* baked *cake*
33.	*girl’s* baking *cake*
34.	*girl’s* cake bake*
35.	*girl’s* cake-bake*
36.	*girl * bake *cake*
37.	*girl * baked *cake*
38.	*girl * baking *cake*
39.	*girl * cake bake*
40.	*girl * cake-bake*
41.	*littl* bake *cake*
42.	*littl* baked *cake*
43.	*littl* baking *cake*
44.	*littl* cake bake*
45.	*littl* cake-bake*
46.	*young* bake *cake*
47.	*young* baked *cake*
48.	*young* baking *cake*
49.	*young* cake bake*
50.	*young* cake-bake*
51.	*cake* baked* by *child*
52.	*cake* baked* by *kid*
53.	*cake* baked* by *boy*
54.	*cake* baked* by *girl*
55.	*cake* baked* by *littl*
56.	*cake* baked* by *young*

Fig. 1. Patterns of a rule <kidback-0> in InfoChat^{TM}. The ’*’ represents a wildcard that may match against characters, words or multiple words.

Each rule has a unique rule identification to distinguish it from other rules and is assigned with an activation value to indicate its current status. A rule consists of a set of stimulus patterns as well as response patterns.

In the compilation of a useful knowledge base, all possible stimulus patterns for a rule must be produced and included in the pattern set. Due to the flexibility of natural languages, the stimulus pattern sets are usually very long and frequently contain many omissions (of realistic user utterances, for which the response would be valid). Figure 2 shows the stimulus pattern set for a rule describing the baking of cakes by children from InfoChat^{TM} (Michie 2001).

There are 56 patterns in the pattern set of this rule. Although the above pattern list is very long, it is clear that the list can be expanded with many more word patterns that have similar meaning to those presented. Thus, the compilation of a stimulus pattern set is a very time-consuming and laborious process with no way of proving the completeness for a set. The requirement for such an exhaustive pattern set is because the conversational agent is using a simple pattern matching scheme, without comprehending the meaning of the user's utterance.

Moreover, using the pattern set of Fig.1, may cause some unintended firing of the rule. For example consider the following sentences:

T1: My little boy loves baking cakes.
T2: All girls like to bake cakes.
T3: Girls like going to cake-bakes.
T4: Some boys enjoy baking cakes.
T5: When I was a boy I occasionally baked cakes.
T6: This little baked cake is inedible.
T7: Skidding cars bake my cakes every time.
T8: On the little hill it was baking hot and the boy was caked in mud.

Fig.2, Some sentences that match patterns in Fig.1.

	T1	T2	T3	T4	T5	T6	T7	T8
T1	1	0.894	0.892	0.822	0.508	0.444	0.620	0.496
T2	0.894	1	0.948	0.848	0.519	0.465	0.591	0.543
T3	0.892	0.948	1	0.863	0.542	0.234	0.529	0.581
T4	0.822	0.848	0.863	1	0.514	0.429	0.637	0.474

Table 1. Similarity between sentences of Fig. 2

The author of rule <kidback-0> would expect sentences T1-4 to fire the rule, but not T5-8. Unfortunately all utterances T1-8 would match stimulus patterns in Fig.1, and fire the rule. As a result, unexpected sentences T5-8 would cause unwanted matches and wrongly fire the rule. This is because the pattern matching scheme merely takes account of the surface information of word patterns appearing in possible user utterances.

A solution for this problem is to equip conversational agents with a matching algorithm that is based on sentence meaning similarity. As presented in the previous sections, the proposed method for sentence similarity is built on the semantic information in the sentences. Unlike simple pattern matching schemes, it does not necessarily require sentences with the same meaning to contain the same words. For example, we may use a single constituent of 'child' to represent *child*, *kid*, *boy*, *girl* and their inflectional forms. This would significantly reduce the size and complexity of the stimulus pattern set necessitating only a small number of example natural language sentences. This overcomes the problem of omissions in the stimulus patterns and massively reduces the workload on the human rule author as the need for suitably placed wildcards and permitted word permutations and substitutions become defunct. Therefore the experiment in this section is to investigate how the proposed sentence similarity method can be applied to conversational agent in the construction of a knowledge base using example human utterances directly.

For sentences in Fig.2, rule authors (scripters) consider T1-4 as expected sentences, while T4-8 are not. The similarity results between these sentenecs are listed in Table 1.

It is observed that similarities in Table 1 can be clearly distinguished in two groups, one group is presented with a shaded background. Similarities between any two sentences in T1-4 have large values, while similarities from a sentence in T1-4 to a sentence in T5-8 are relatively small. By introducing a similarity threshold in the firing scheme, we can prevent the unwanted matches from T5-8. This indicates that it is possible to use a single sentence to represent T1-4. Thus we can reasonably use a sentence from T1-4 to replace the 56 patterns of Fig.1.

Taking the above observations into account, we are incorporating the proposed similarity method into a conversational agent. The scripter devises one or only a few stimulus sentence(s) conveying the meaning of expected user utterances for each rule. The sentences are in the form of natural language and stored in the knowledge base. During the execution of the conversational agent, the user's utterance is received by the agent. The agent computes the similarity between the user's utterance and the stimulus sentences in the knowledge base, using the proposed sentence similarity algorithm. The similarity is further converted to a firing strength using a strategy as described in InfoChat^{TM} (Michie 2001). The rule with the greatest firing strength is then fired.

In comparison to simple pattern matching algorithms, the immediate benefits of incorporating sentence similarity is obvious, in that the rule is much shorter, more readable and hence far easier to maintain. However, this does not mean that it will completely remove the need for pattern matching schemes from conversational agents. Rather the proposed similarity method can form a complement to existing pattern matching schemes. Pattern matching schemes may be more reliable for irregular rules which match against grammatically incorrect (more similar to certain language) user utterances. Therefore, the agent's knowledge base may contain two distinct sets of stimulus patterns, natural language sentences and (when appropriate) word patterns with wildcards.

4 Conclusion

This paper presented a method for measuring sentence similarity. The method computes sentence similarity from semantic information and word order information shared by the concerned sentences. Firstly, semantic similarity is derived from a lexical knowledge base and corpus. A lexical database describes common human knowledge about words in a natural language, this knowledge is usually stable across a wide range of language application areas. The corpus represents the actual usage of language and words. Thus our semantic similarity not only captures common human knowledge, but is also able to adapt to a specific application. This adaptation is achieved by using information from an application specific corpus. Secondly, the proposed method considers the impact of word order in sentences. The derived word order similarity measure takes into account the number of different words as well as the number of word pairs in different order. The overall sentence similarity is then defined as a combination of semantic similarity and word order similarity. In accordance with the view that word order plays a subordinate role for interpreting sentences, we weight word order similarity less in defining the overall sentence similarity. To investigate the value of the proposed method in real applications, it was applied to a conversational agent to simplify the agent's knowledge representation and processing. A strategy for incorporating a pattern matching scheme and sentence similarity was proposed. This results in a conversational agent knowledge base that is easier to compile, far shorter, more readable and much easier to maintain.

References

Allen, J. 1995 *Natural Language Understanding*. Benjamin Cummings, Redwood City

Bates, M. 1986 Subject Access in Online Catalogue: a Design Model, *J. American Society for Information Science* 11: 357-376.

British National Corpus home page: http://www.hcu.ox.ac.uk/BNC/.

Burgess, C., Livesay, K., and Lund, K. 1998. Explorations in Context Space: Words, Sentences, Discourse. *Discourse Processes* 25: 211-257

Foltz, P.W., Kintsch, W. and Landauer, T.K. 1998. The Measurement of Textual Coherence with Latent Semantic Analysis. *Discourse Processes* 25 285-307

Wiemer-Hastings, P. 2000. Adding Syntactic Information to LSA. In *Proceedings of the twenty-second Conference on Cognitive Science*, 989-993 Mahwah, NJ: Lawrence Erlbaum Associates

Hatzivassiloglou, V., Klavans, J., Eskin, E.1999. Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning. Empirical Methods in natural Language Processing.

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

Kozima, H.1994 Computing Lexical Cohesion as a Tool for Text Analysis. PhD. Diss., Graduate School of Electro-Communications, Univ. of Electro-Communications

Landauer, T.K., Laham, D., Rehder, B.and Schreiner, M.E. 1997 How Well can Passage Meaning be Derived without Using Word Order? A Comparison of Latent Semantic Analysis and Humans. In *Proceedings of the Nineteenth Meeting Cognitive Science.* Erlbaum, Mawhwah: 412-417

Li, Y.H., Bandar, Z. and McLean, D. 2003 An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. IEEE Transactions on Knowledge and Data Engineering . Volume 15 Number 4., pp871-882.

McHale, M. 1998. A Comparison of WordNet and Roget's Taxonomy for Measuring Semantic Similarity. Proc. COLING/ACL Workshop Usage of WordNet in Natural Language Processing Systems. Montreal.

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

Michie, D. 2001 Return of the Imitation Game, *Electronic Transactions on Artificial Intelligence* 6 (2001)

Miller, G.A. 1995 WordNet: a Lexical Database for English. Communications of the ACM 38:39-41

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