A hybrid Web-based measure for computing semantic relatedness between words

Gerasimos Spanakis
gspana@ece.ntua.gr

Georgios Siolas
gsiolas@image.ntua.gr

Andreas Stafylopatis
andreas@cs.ntua.gr

Intelligent Systems Laboratory
National Technical University of Athens
Iroon Polytechneiou 9, 15780, Zografou
Athens, Greece

Abstract

In this paper, we build a hybrid Web-based metric for computing semantic relatedness between words. The method exploits page counts, titles, snippets and URLs returned by a Web search engine. Our technique uses traditional information retrieval methods and is enhanced by page-count-based similarity scores which are integrated with automatically extracted lexico-synantic patterns from titles, snippets and URLs for all kinds of semantically related words provided by WordNet (synonyms, hypernyms, meronyms, antonyms). A support vector machine is used to solve the arising regression problem of word relatedness and the proposed method is evaluated on standard benchmark datasets. The method achieves an overall correlation of 0.88, which is the highest among other metrics up to date.

1 Introduction

Semantic relatedness between words has always been a challenging problem in data mining. Nowadays, World Wide Web (WWW) has become a huge warehouse of data and documents, with available information for every single user query. Web search engines retrieve the most relevant documents, according to the word(s) we provide them, and the content of the retrieved documents is more or less related to the common senses of the word(s) given.

We propose a method to compute semantic relatedness between words based on Web search engines, using the kinds of semantic relationship provided by the structured lexical database WordNet [6] (this means: synonyms, hyper/hyponyms, antonyms and meronyms) along with traditional information retrieval techniques. We consider the three main components of a usual search result structure (title, snippet, URL), based on the idea that similar words would appear in the same pages in any of the above components.

The remainder of the paper is organized as follows. In Section 2 we discuss related work dealing with semantic similarity computation between words (both hand-crafted and statistical methods). The proposed method is fully described in Section 3. The conducted experiments and the benchmark datasets with the evaluation results are presented in Section 4. In Section 5 we conclude the paper and give directions for further improvement of our method.

2 Related Work

Semantic relatedness measures are used in many applications such as word-sense disambiguation [18], query expansion [25], Web-page annotation [4], hence many measures have been proposed.

Hand-crafted lexical databases such as WordNet [6] encode relations between words. Until now, many metrics have been defined computing relatedness using various properties of its database structure ([17], [14], [12], [11], [9], [2]). The fact that these methods are based on hand-crafted language resources such as ontologies or thesauri, brings up several drawbacks such as: time and effort to build and maintain a repository, absence of proper names, neologisms, technical terms e.t.c.

On the other hand, Web pages have become a great source of information for various similarity tasks. Turney [24] used point-wise mutual information and information retrieval from Web search to recognize synonyms. Sahami et al. [19], computed semantic similarity between two queries using snippets returned for those queries by a Web search engine. For each query, they collect snippets from a Web search engine and represent each snippet as a $TF - IDF$-weighted term vector [20]. Semantic similar-

ity between two queries is then defined as the inner product between the corresponding centroid vectors.

Chen et al. [3], proposed a double-checking model using text snippets returned by a Web search engine to compute semantic similarity between words. For two words P and Q, they collect snippets for each word from a Web search engine. They proceed by counting the occurrences of word P in the snippets for word Q and the occurrences of word Q in the snippets for word P. These values are then combined nonlinearly to compute the similarity between P and Q.

Iosif et al. [10] came up with an unsupervised model which, in order to compute the similarity between two words P and Q, downloads a number of the top ranked documents for the query P AND Q and applies "wide-context"and "narrow-context"metrics.

Bollegala et al. [1] proposed a method which exploits page counts and text snippets returned by a Web search engine. They define various similarity scores for two given words P and Q using the page count for queries P, Q and by extracting lexico-syntactic patterns from the text snippets returned by the Web search engine.

3 Method

We propose a statistical method which uses traditional information retrieval techniques and also takes into account all kinds of relationship between two words and decides upon their relatedness. First of all, we apply the "Bag of Words" representation [21] to the Web search results text for each word and compute an early metric (Rel_{BOW}) for word relatedness. Then, we enhance our method by using four popular co-occurrence measures to calculate page-count-based similarity metrics for the pairs of the words and by automatically extracting lexico-syntactic patterns about the pairs of the words based on the title, snippet and URL of the Web search results. We then train regression support vector machines (SVMs) to fetch the relatedness of word pairs leading to another metric(Rel_{SVM}), which is integrated with the (Rel_{BOW}). For evaluation purposes we applied our approach on standard datasets used in the literature. More specifically, we use the Similarity-353 dataset [7] for training the SVM excluding the 28 pairs of the Miller-Charles dataset [16] which are used for testing and for the "Bag of Words" model.

A linear combination of the measure computed with the "Bag of Words" model (Rel_{BOW}) and the measure provided by the regression support vector machine (Rel_{SVM}), allows to quantify the overall relatedness Rel_{total} between two words:

$$Rel_{total} = \lambda Rel_{SVM} + (1 - \lambda) Rel_{BOW} \qquad (1)$$

where $\lambda \in (0, 1)$ is a parameter to weight the contribution of each method to the hybrid measure.

3.1 The "Bag of Words" model for computing the Rel_{BOW} measure

The traditional document representation is a word-based vector (Bag Of Words, or BOW), where each dimension is associated with a term of the dictionary which contains all the words appearing in the corpus [21]. The value associated with a given term reflects its frequency of occurrence within the corresponding document (Term Frequency, or tf), and within the entire corpus (Inverse Document Frequency, or idf). tf can be computed using the following equation:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \qquad (2)$$

where $n_{i,j}$ is the number of occurences of the considered term t_i in document d_j and the denominator is the sum of number of occurences of all terms in document d_j.

Similarly, idf is computed using the following equation:

$$idf_i = \log \frac{N}{df_i} \qquad (3)$$

where N is the total number of documents and df_i represents the number of documents which contain the term t_i. Then, the weight of a term i in a document j is computed using the following equation:

$$(tf - idf)_{i,j} = tf_{i,j} \cdot idf_i \qquad (4)$$

where a high value is reached by a high term frequency (in the given document) and a low frequency of the term in the whole collection of documents. The weights hence tend to filter out common terms.

In our approach, we consider a dataset (S) consisting of words (w). We notice that Web search results are organized in fragments containing three parts : the title of the Web page, the text snippet and the URL of the indexed page. These three parts contain valuable information about the context of a word. We aim at representing each word w with a document d, which is substantiated by the Web search results for the query for this word w.

Specifically, we retrieve the first 1000 Web search results for the query of word w and we join all titles and snippets into one document (d). So now, every word of the dataset is represented by such a document, to which we can apply traditional preprocessing techniques (eliminating stop words, stemming, normalization). A set of documents is thus obtained, where each document d corresponds to a word w. After this procedure, if we pick all the different terms (t_i) appearing in all documents (d), we build the dictionary $(dict)$ of our dataset. We choose to omit those terms with the quantity $\sum_{j=0}^{N}(tf - idf)_{i,j}$ lower than a threshold. Equation (4) is used, where i denotes the terms t_i, j denotes the documents d and N is the total number of documents. We can alternatively choose to include all terms t_i.

For every document d (thus every word w) we now create a vector $\mathbf{v}(w)$, where each component i of this vector is set to the $tf - idf$ value (as computed by equation 4) of the term t_i for the specific document. Consequently, we end up with $|S|$ vectors (where $|S|$ is the number of words of our dataset).

The association between two words w_1, w_2 is then measured by computing the Dice coefficient between the corresponding vectors using the following equation:

$$Rel_{BOW}(w_1, w_2) = \frac{2 \cdot \mathbf{v}(w_1) \cdot \mathbf{v}(w_2)}{||\mathbf{v}(w_1)||^2 + ||\mathbf{v}(w_2)||^2} \quad (5)$$

where : $\mathbf{v}(w_1) \cdot \mathbf{v}(w_2)$ stands for the dot product of the two vectors and $||\mathbf{v}||$ stands for the euclidian norm of vector $\mathbf{v}$. The algorithm is described in Fig. 1.

"Bag of Words" model for computing Rel_{BOW}

1: $ListOfTerms_{tot} \leftarrow NULL$
2: **for all** words w from dataset S **do**
3: - join Titles and Snippets from the first 1000 results of $query(w)$ into a document d
4: - apply stop-words elimination, stemming, normalization
5: - get the list of terms $ListOfTerms_w$ which represent the document d (thus the word w) and compute the tf using equation (2)
6: **end for**
7:
8: **for all** different terms t_i appearing in all $ListOfTerms_w$ **do**
9: - compute idf using equation (3)
10: - compute $tf - idf_i$ using equation (4)
11: **if** $\sum_{j=0}^{N} tf - idf_{i,j} \geq threshold$ **then**
12: - Append t_i to $ListOfTerms_{tot}$
13: **end if**
14: **end for**
15:
16: - $ListOfTerms_{tot}$ will be used to form vector v
17: **for all** words w from dataset S **do**
18: - form and fill the $\mathbf{v}(w)$ which represents the word w
19: **end for**
20:
21: **for** words w_1 and w_2 **do**
22: - compute Rel_{BOW} metric according to equation (5)
23: **end for**

Figure 1. Description of Rel_{BOW} measure

This method has three major drawbacks: (1) it breaks multi-word expressions into completely independent terms, (2) it considers polysemous words as one single component and (3) it maps synonymous words into different compounds, thus, when applied by its own, has limited effectiveness. It is therefore essential to further embed semantic information and conceptual patterns so as to be able to enhance the prediction capabilities of word relatedness.

3.2 Feature Extraction from Web search results

3.2.1 Page-count-based similarity metrics

The basic idea supporting these metrics is that word co-occurrence in Web results is likely to indicate semantic similarity. Word co-occurrence is expressed through the page counts of query P *AND* Q (let P and Q be a pair of words). However, it is reasonable to take into account not only the page counts of query P *AND* Q, but also the number of documents that include each word (P or Q) individually.

We consider four popular co-occurrence measures, which express semantic similarity using page counts. The following notation for the page counts of a Web search engine will be adopted: $H(P)$ denotes the page counts for query P, $H(Q)$ denotes the page counts for query Q, and $H(P \bigcap Q)$ denotes the page counts for query P AND Q.
The Jaccard coefficient for a pair of words (P and Q) is defined as:

$$Jaccard(P, Q) = \frac{H(P \bigcap Q)}{H(P) + H(Q) - H(P \bigcap Q)} \quad (6)$$

The Overlap coefficient is defined as:

$$Overlap(P, Q) = \frac{H(P \bigcap Q)}{\min(H(P), H(Q))} \quad (7)$$

The Dice coefficient is defined as:

$$Dice(P, Q) = \frac{2 \cdot H(P \bigcap Q)}{H(P) + H(Q)} \quad (8)$$

Finally, the PMI (point-wise mutual information) is defined as:

$$PMI(P, Q) = \log \frac{\frac{H(P \bigcap Q)}{N}}{\frac{H(P)}{N} \cdot \frac{H(Q)}{N}} \quad (9)$$

where N is the number of documents indexed by the Web search engine. In our experiments we set $N = 10^{10}$, according to the number of indexed pages reported by Google.

3.2.2 Lexico-syntactic pattern extraction using title, snippet and URL

Consider the following Web search results example for the query *"cats AND dogs":*

Title: Friends of Cats & Dogs - Home
Snippet: The Friends of Cats and Dogs Foundation (Los Amigos de Gatos y Perros Foundacion) seeks to help stray animals, pets, and their owners through various ...
URL: www.fcdf.org/cats-and-dogs.html

We notice that both words of the query (cats, dogs) appear in the title, snippet and the URL with the following

patterns: *"Cats & Dogs"* for the title, *"Cats and Dogs"* for the snippet and *"cats-and-dogs"* for the URL. If we replace the query words with the wildcards X and Y respectively, we extract a pattern *"X & Y"* for the title, a pattern *"X and Y"* for the snippet and a pattern *"X-and-Y"* for the URL.

At this point, we should mention that, by considering all kinds of semantic relationships present in WordNet (synonyms, hypernyms, meronyms, antonyms), we have the ability to extract numerous such patterns for pairs of words depending on the kind of word relationship. However, as far as this method is concerned, we implemented the algorithm shown in (Fig.2) to extract lexico-syntactic patterns, extending the algorithm of [1], by taking into account not only the snippets, but also the titles and the URLs.

Pattern Extraction from Web search results

1: **for all** wordpairs (A, B) from all classes (synonyms, hypernyms, meronyms, antonyms, non-related words) **do**
2: $T \leftarrow$ GetAllTitles($query(A, B)$)
3: $S \leftarrow$ GetAllSnippets($query(A, B)$)
4: $U \leftarrow$ GetAllURLS($query(A, B)$)
5: $NT \leftarrow 0$
6: $NS \leftarrow 0$
7: $NU \leftarrow 0$
8: **end for**
9:
10: **for all** titles t from T **do**
11: $NT \leftarrow NT$ + GetPatterns(t, A, B)
12: $TitlePatterns \leftarrow$ CountFrequency NT
13: **end for**
14: **return** $TitlePatterns$
15:
16: **for all** snippets s from S **do**
17: $NS \leftarrow NS$ + GetPatterns(s, A, B)
18: $SnippetPatterns \leftarrow$ CountFrequency NS
19: **end for**
20: **return** $SnippetPatterns$
21:
22: **for all** urls u from U **do**
23: $NU \leftarrow NU$ + GetPatterns(u, A, B)
24: $URLPatterns \leftarrow$ CountFrequency NU
25: **end for**
26: **return** $URLPatterns$

Figure 2. Pattern extraction algorithm

Given a set T of word-pairs, we get the Google results for this word pair and then process separately the title, the snippet and the URL. For each one of these parts and for all the Web search results, we keep those parts that contain both words. We then examine the "context "occurring between these two words and omit those examples with "context"containing more than 4 words and/or words other than an extended list of common stop-words (procedure $GetPatterns$ in Fig. 2). (The rest of the words are too specific for the word-pair). We compute the frequency of each pattern and build a list of the most common patterns in the title, snippet and URL for Web search results.

It is obvious that, in order to get the most important patterns, a big enough dataset has to be used. In order to extract patterns for every WordNet relation we use five different classes each one containing around 1800 pairs of synonyms, 1800 pairs of hypernyms, 1800 pairs of meronyms, 1800 pairs of antonyms and 1800 pairs of non-related words respectively, all derived from WordNet. By processing the Web search results for the totality of word pairs, we end up with numerous different patterns describing the four relations mentioned above.

3.3 Feature selection for selecting the most valuable patterns

Apparently, it is not feasible to use a very large number of patterns to train a regression SVM (or any other model) in order to decide upon the word relatedness. We have to use a feature selection algorithm in order to choose the patterns having the highest discriminative power.

We use the idea of multi-class feature selection [8] in order to determine which features are most important for each class of the four semantic categories. The pattern selection algorithm is described in Fig. 3.

Pattern Selection for choosing the most valuable patterns

1: **for all** classes c of synonyms, hypernyms, antonyms, meronyms **do**
2: - rank all features according to criterion M for the subtask of discriminating class c vs. all other classes combined
3: - store this feature ranking for class c
4: **end for**
5:
6: **while** list of features ($list$) not complete **do**
7: - call Round-Robin scheduler to select next class c
8: - select the next feature f from M-ranking for class c
9: - append f to the $list$ if not already present
10: **end while**

Figure 3. Pattern selection algorithm

Our goal is to rank the patterns by their ability to discriminate each initial class (synonyms, hypernyms, antonyms, meronyms), which from now on will be called positive class, against another class (which will be called negative), consisting of all the other classes combined (i.e. if the synonyms are the positive class, then the negative class will contain equal number of hypernyms, antonyms and meronyms). To achieve this we define a criterion called M. For each extracted pattern we create a contingency table, as shown in Table 1. In this table, A denotes pattern frequency in the positive class examined, B denotes the pattern frequency in the negative class containing pairs of all other classes, U denotes the total frequency of all patterns for the positive class examined and V denotes the total frequency of all patterns for the negative class.

Table 1. Contingency table for pattern x

	Pattern x	Other than x
Freq. for the positive class examined	A	C=U-A
Freq. for the negative class examined	B	D=V-B

We define the (Pointwise) Mutual Information (MI) and Information Gain (IG) [15] measures for our method as follows:

$$MI = \log \frac{\frac{A}{N}}{\frac{(A+B)}{N} \cdot \frac{(A+C)}{N}} \quad (10)$$

$$IG = e(A+C, B+D) - [\frac{A}{N} \cdot e(A, B) + \frac{C}{N} \cdot e(C, D)] \quad (11)$$

where

$$e(x, y) = -\frac{x}{x+y} \cdot \log \frac{x}{x+y} - \frac{y}{x+y} \cdot \log \frac{y}{x+y} \quad (12)$$

and we define a novel pattern importance measure (PI):

$$PI = \begin{cases} IG, \text{ if } & MI > 0 \\ 0, \text{ if } & MI \leq 0 \end{cases} \quad (13)$$

The pattern importance measure omits patterns whose MI is negative, because negative MI implies (through Equation 10) logarithm argument value less than 1, which in turn means that the denominator is greater than the numerator. Through our experiments we noticed that the meaning of this is that either B and/or C is a large number, an indication that the pattern is more frequent in classes other than the one examined.

Using Equation (13) we create feature rankings for every class and then –through a round-robin process– we choose the most discriminative (distinct) features that discriminate each class (omitting the class of non-related words) from the others. A threshold is used, which is set by the user, depending on how big we want the feature vector to be. In Table 2 we present the most discriminative patterns for title, snippet and URL. The top-3 patterns are different for titles, snippets and URLs, which implies that there is information lying under the Web search results structure.

Table 2. Top-3 highly discriminative patterns

Top-3 patterns		
Titles	Snippets	URLs
X or Y	X, Y	X-or-Y
X in Y	X for Y	X+Y
X (Y	X in Y	X/Y

3.4 Creating the feature vector

We construct the feature vector for each pair of words (P and Q) according to the following procedure. We collect the Google results for the query *"P AND Q"* and separate the titles, snippets and URLs. According to the procedure described in Section 3.2.1 we get the page-count similarity metrics and then integrate those metrics with the features generated through the process described in Section 3.3. By this procedure we form a feature vector of maximum dimension 811 (807 lexico-synantic patterns and 4 similarity metrics). By selecting a different threshold of pattern importance in the round-robin procedure, we can control the vector dimension. Using these feature vectors for each word pair we train a regression support vector machine with the labeled feature vectors of the Similarity-353 dataset (excluding the 28 pairs of Miller-Charles used for testing). Once we have trained the regression-SVM, we are in a position to use the system for computing semantic relatedness between any two given words. We only need to create a feature vector for the word pair and the SVM will compute the relatedness measure.

3.5 Regression SVM for computing the R_{SVM} measure

In a regression support vector machine (rSVM)[5], the objective is to estimate the functional dependence of variable y on a set of indepedent variables x (which constitute the input space, denoted as X). Like in other regression problems, the starting hypothesis is that the relationship between the independent and dependent variables is given by a deterministic function plus some noise ϵ:

$$y = f(\mathbf{x}) + \epsilon \quad (14)$$

The task is then to find a functional form for f that can correctly predict new cases not previously presented to the SVM. This can be achieved by training the SVM model on a sample set (training set), a process that involves the sequential optimization of an error function:

$$error = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=0}^{N}\xi_i + C\sum_{i=0}^{N}\xi_i' \quad (15)$$

where C is the capacity constraint (trade-off between training error and margin), $\mathbf{w}$ is the vector of coefficients ($\mathbf{w} \in X$), i is an index labeling the N training cases and ξ_i, ξ_i' are parameters for handling non-separable data.

Our goal is to minimize the error, subject to :

$$\mathbf{w}^T \cdot \phi(x_i) + b - y_i \leq \epsilon + \xi_i' \quad (16)$$

$$y_i - \mathbf{w}^T \cdot \phi(x_i) - b \leq \epsilon + \xi_i \quad (17)$$

$$\xi_i', \xi_i \geq 0 \quad (18)$$

where b is a constant ($b \in \Re$) and $\phi(x_i)$ is a kernel function (linear, polynomial, RBF, sigmoid) used to transform data from the input data (indepedent) to the feature space.

We trained an rSVM [13] with an RBF function kernel. The model was optimized with respect to parameters C and γ (the γ coefficient of the radial basis function). We experimented with other kernel types (linear, polynomial) as well, but best performance was achieved using radial basis functions. All the results reported in the next section were achieved by an RBF kernel.

4 Experiments

We evaluated the proposed method by comparing our results against the Miller-Charles benchmark dataset. The Miller-Charles dataset consists of 28 word-pairs [16] rated by a group of 38 human subjects. The word pairs are rated on a scale from 0 (no similarity) to 4 (perfect synonymy). As a training set for the rSVM we used the Similarity-353 dataset [7], excluding the 28 pairs of the Miller-Charles dataset. At this point we must highlight the problem of not having a large and reliable dataset of semantically related (and unrelated) words. WordNet provides us with many word pairs and using a measure from those described in Section 2, one can produce many word pairs along with their similarity metric. Two problems arise here. First, the measures described in Section 2 do not constitute an absolutely correct metric for computing similarity due to the fact that they are based on an ontology, thus we expect deviations from human results. Second, WordNet contains lots of words which are not good examples for getting Web results or are just too farfetched.

For the Rel_{BOW} measure, we obtained the first 1000 Web search results for the different words of Miller-Charles dataset and then by the procedure described in Fig. 1, we computed the relatedness between the word pairs of the dataset. Each word was represented by a document consisting of the titles and snippets from all the first 1000 Web search results returned by Google. Obviously, the vocabulary (each word of the vocabulary is a term of the vector representing the documents) contains all the words from all documents. The drawback of this measure, is that, if a word we want to use in order to compute semantic relatedness does not exist in our vocabulary, we will have to reform the features of the vector, reflecting the changes caused by the new document.

For the Rel_{SVM} measure, we trained the Support Vector Machine for the regression problem, experimenting with its parameters. The optimized parameters for the SVM are presented in Table 3.

We analyzed the behavior of the Rel_{SVM} measure with respect to the number of patterns used as features and the

Table 3. Optimal parameters for the SVM

SVM optimal parameters	
Parameter	Value
Kernel	RBF
$gamma$	0.004
C	7

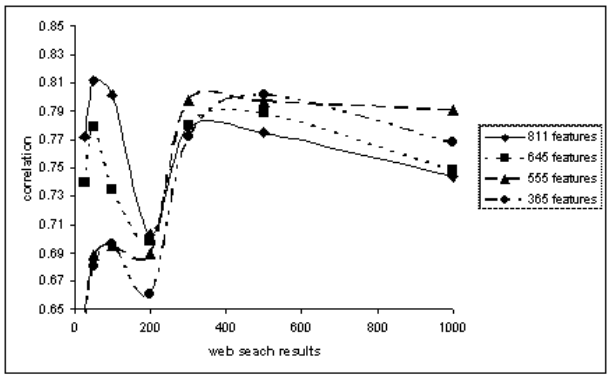

Figure 4. Rel_{SVM} correlation vs. number of Web results and features

number of Web search results. Fig. 4 shows the impact that Web results and number of features have on correlation. We observe that using a small number of Web results and a small number of features yields low correlation values. As we increase the number of features and (or) the number of Web results, we manage to leverage the correlation. Moreover, it seems that, as more Web results and more features are used, the correlation begins to decay, probably beacuse the feature vectors become very sparse. Better correlation is achieved when using a low number of Web results and a high number of features (patterns). This might be due to the fact that, in general, only the the first results of Google contain high quality information about the word pair, but attention is needed so as not to use very few web results because the correlation begins to decay again. Our experiments indicated the use of 50 Web search results, with a feature-rich vector (dimension was 811) by choosing a low threshold in the procedure described in section 3.3 for selecting patterns.

After computing the two individual measures, we experimented on the training set with various values of the λ parameter in equation (1), in order to weigh the two measures. The results presented here, were obtained using $\lambda = 0.7$ and are normalized in the interval $[0, 1]$ for ease of comparison. Table 4 shows the results of our method in comparison to other methods and Table 5 displays a comparison of the characteristics of various methods.

All of the methods were evaluated with respect to the 28 pairs of the Miller-Charles dataset. The Jiang measure [12] exploits the semantically hierarchical structure of

Table 4. Word similarity methods results comparison

Measures	miller-charles	jaccard	dice	overlap	PMI	Sahami	CODC	SemSim	Jiang	Binary CS	Proposed
cord-smile	0.13	0.06	0.07	0.08	0.03	0.09	0	0	0.35	0.4	0.23
rooster-voyage	0.08	0.02	0.02	0.03	0.04	0.20	0	0.02	0.08	0	0.29
noon-string	0.08	0.13	0.14	0.11	0.30	0.08	0	0.02	0.18	0.16	0.32
glass-magician	0.11	0.08	0.08	0.27	0.37	0.14	0	0.18	0.68	0.18	0
monk-slave	0.55	0.22	0.23	0.12	0.71	0.10	0	0.38	0.39	0.19	0.32
coast-forest	0.42	0.92	0.92	0.32	0.55	0.25	0	0.41	0.29	0.76	0.45
monk-oracle	1.1	0.07	0.08	0.07	0.32	0.05	0	0.33	0.34	0.47	0.31
lad-wizard	0.42	0.07	0.08	0.05	0.33	0.15	0	0.22	0.32	0.37	0.31
forest-graveyard	0.84	0.10	0.11	0.35	0.59	0	0	0.55	0.19	0.11	0.19
food-rooster	0.89	0.04	0.05	0.39	0.28	0.08	0	0.06	0.4	0.35	0.45
coast-hill	0.87	1	1	0.40	0.53	0.29	0	0.87	0.71	0.18	0.42
car-journey	1.16	0.37	0.39	0.43	0.22	0.19	0.29	0.29	0.33	0.52	0.44
crane-implement	1.68	0.22	0.24	0.13	0.61	0.15	0	0.13	0.59	0.1	0.25
brother-lad	1.66	0.16	0.17	0.33	0.52	0.24	0.38	0.34	0.28	0.58	0.41
bird-crane	2.97	0.24	0.26	0.27	0.57	0.22	0	0.88	0.73	0.59	0.60
bird-cock	3.05	0.33	0.35	0.23	0.53	0.06	0.50	0.59	0.73	0.44	0.53
food-fruit	3.08	0.91	0.91	1	0.57	0.18	0.34	1	0.63	0.79	0.79
brother-monk	2.82	0.18	0.19	0.37	0.55	0.27	0.55	0.38	0.91	0.63	0.47
asylum-madhouse	3.61	0.07	0.08	0.13	1	0.21	0	0.77	0.97	0.51	0.76
furnace-stove	3.11	0.33	0.35	0.14	0.96	0.31	0.93	0.89	0.39	1	0.91
magician-wizard	3.5	0.25	0.27	0.23	0.78	0.23	0.67	1	1	0.59	0.94
journey-voyage	3.84	0.50	0.52	0.21	0.52	0.52	0.42	1	0.88	0.75	0.89
coast-shore	3.7	0.88	0.89	0.53	0.71	0.38	0.52	0.95	0.99	0.5	0.61
implement-tool	2.95	0.50	0.52	0.55	0.53	0.42	0.42	0.68	0.97	0.8	0.72
boy-lad	3.76	0.18	0.19	0.52	0.54	0.47	0	0.97	0.88	0.67	0.75
automobile-car	3.92	0.55	0.57	0.69	0.37	1	0.69	0.98	1	0.76	1
midday-noon	3.42	0.18	0.20	0.25	0.94	0.29	0.86	0.82	1	0.74	0.92
gem-jewel	3.84	0.37	0.39	0.15	0.68	0.21	1	0.69	1	0.53	0.95
Correlation	**1**	**0.27**	**0.28**	**0.38**	**0.62**	**0.58**	**0.69**	**0.83**	**0.83**	**0.71**	**0.88**

WordNet and achieves the best correlation among ontology-based methods, with the drawbacks of these techniques (as they were described in Section 2). Among Web search results based methods, the page-count based metrics achieve the lowest correlations, with PMI being the most accurate. The Sahami [19] and Iosif [10] measures (unsupervised) achieve a moderate correlation and the Bollegala [1] metric appears to have the second best performance using only synonyms. The method proposed here achieves a correlation 0.88, which is the highest among all other metrics up to date.

5 Conclusion and future work

In this paper we use Web search results to build a Web-based relatedness measure between pairs of words. The proposed measure exploits all the information of Web search results (title, snippet, URL) combining traditional information retrieval techniques ("Bag of words" model) with page-count-based similarity metrics along with the extraction of lexico-syntactic patterns from the results. We took into account all kinds of relationship provided by WordNet (synonyms, hypernyms, meronyms, antonyms) and the Web search results structure information for extracting a measure of relatedness. We introduced a new measure for feature selection combining mutual information and information gain and trained a regression classifier model (SVM). The SVM-based measure(Rel_{SVM}) and the "Bag of words" model (Rel_{BOW}) are combined together to a robust measure. The proposed method achieves the highest correlation value on the Miller-Charles dataset compared to the other state-of-the-art metrics.

We are currently investigating ways to improve the proposed measure. We look into a method of creating a reliable dataset of related words in order to achieve better training. In addition, further research is needed to explore ways of overcoming the drawbacks of Rel_{BOW}, thus creating an even more robust metric. Moreover, we will try to exploit the information lying in the structure of web search results using more complex techniques like in [23].

References

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

[2] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Comput. Linguist.*, 32(1):13–47, 2006.

[3] H.-H. Chen, M.-S. Lin, and Y.-C. Wei. Novel association measures using web search with double checking. In *ACL-*

Table 5. Characteristics of different semantic similarity calculation methods

	Web results	page counts	snippets	Web results structure	patterns	wordnet	document download	traditional IR techniques	correlation
Jaccard	X	X							0.24
Dice	X	X							0.25
PMI	X	X							0.63
Overlap	X	X							0.30
Sahami	X		X					X	0.58
CODC	X		X						0.69
SemSim	X	X	X		X	X			0.83
Jiang						X			0.83
CS(W/WS)	X						X	X	0.71
Proposed	X	X	X	X	X	X		X	0.88

44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pages 1009–1016, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[4] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 462–471, New York, NY, USA, 2004. ACM.

[5] R. Collobert, S. Bengio, and C. Williamson. Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[6] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May 1998.

[7] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, 2002.

[8] G. Forman. A pitfall and solution in multi-class feature selection for text classification. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 38, New York, NY, USA, 2004. ACM.

[9] G. Grefenstette. Sextant: Exploring unexplored contexts for semantic extraction from syntactic analysis. In *Proceedings of the 30st annual meeting of the Association for Computational Linguistics, ACL*, pages 324–329, 1992.

[10] E. Iosif and A. Potamianos. Unsupervised semantic similarity computation usingweb search engines. In *WI '07: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 381–387, Washington, DC, USA, 2007. IEEE Computer Society.

[11] M. Jarmasz. Roget"s thesaurus as a lexical resource for natural language processing. Technical report, University of Ottawa, 2003.

[12] J. J. Jiang and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *International Conference Research on Computational Linguistics (ROCLING X)*, pages 9008+, September 1997.

[13] T. Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169–184, 1999.

[14] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[15] C. D. Manning and H. Schútze. *Foundations of Statistical Natural Language Processing*. The MIT Press, June 1999.

[16] G. A. Miller and W. G. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.

[17] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–45, 1995.

[18] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[19] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 377–386, New York, NY, USA, 2006. ACM.

[20] G. Salton and M. J. Mcgill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[21] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[22] F. Sebastiani and C. N. D. Ricerche. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47, 2002.

[23] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 104, New York, NY, USA, 2004. ACM.

[24] P. D. Turney. Mining the web for synonyms: PMI–IR versus LSA on TOEFL. *Lecture Notes in Computer Science*, 2167:491–502, 2001.

[25] B. Vélez, R. Weiss, M. A. Sheldon, and D. K. Gifford. Fast and effective query refinement. *SIGIR Forum*, 31(SI):6–15, 1997.