Computing Semantic Similarity Using Large Static Corpora

András Dobó, János Csirik

University of Szeged, Institute of Informatics, Szeged, Hungary
{dobo,csirik}@inf.u-szeged.hu

Abstract. Measuring semantic similarity of words is of crucial importance in Natural Language Processing. Although there are many different approaches for this task, there is still room for improvement. In contrast to many other methods that use web search engines or large lexical databases, we developed such methods that solely rely on large static corpora. They create a binary or numerical feature vector for each word making use of statistical information obtained from the corpora. These vectors contain features based on context words or grammatical relations extracted from the corpora and they employ diverse weighting schemes. After creating the feature vectors, word similarity is calculated using various vector similarity measures. Beside the individual methods, their combinations were also tested. Evaluated on both the Miller-Charles dataset and the TOEFL synonym questions, they achieve competitive results to recent methods.

Keywords. semantic similarity, static corpora, co-occurrence statistics

1 Introduction

For many Natural Language Processing (NLP) tasks, such as information extraction, spelling correction or word sense disambiguation, knowing the semantic similarity of words can be very helpful. Therefore, in the last approximately 20 years, much research has been done on developing methods that can automatically compute the semantic similarity of words. Most of the best performing methods employ web search engines (for example Google or Yahoo!) or large lexical databases (such as WordNet or Roget's Thesaurus) in order to compute word similarity. Although their application can be advantageous for many reasons, and systems using them tend to perform well, they also have many disadvantages.

Using web search engines in NLP tasks can have many drawbacks, as noted by Kilgarriff [1] among others. First, the returned page hit counts are not exact counts and they change over time. Furthermore, queries can have no linguistic restrictions and punctuation cannot be used. Moreover, their use can be limited and time consuming. Finally, they usually have a constraint on the number of pages returned per query.

Employing large lexical databases induce other problems. Methods that can automatically compute the semantic similarity of words are especially useful for uncom-

The original publication is available at www.springerlink.com via
http://dx.doi.org/10.1007/978-3-642-35843-2_42.

mon words not included in lexical databases and thesauri. However, an algorithm that solely relies on lexical databases is not able to compute the similarity of such words, and therefore cannot be used in those cases when the most useful they would be. Furthermore, as languages evolve over time and new words are created every day, these lexical databases should be revised constantly, which is a costly task. Moreover, every manually created database is prone to human errors: important words, word meanings (called synsets in WordNet) and relations can be missing from them.

Summing up, there are situations, where the usage of web search engines or large lexical databases is not suitable or feasible because of the above mentioned problems. In those cases, such methods are needed that use neither web search engines nor lexical databases. Therefore, we constructed methods that solely rely on large static corpora.[1] They first process the used corpora and create a feature vector for each word using context words or grammatical relations as features and some weighting scheme. Then, they compute word similarity based on the similarity of these word vectors. Beside using the created methods by themselves, a number of combination of the different methods were also examined. Tested on two different datasets, namely the TOEFL synonym questions and the Miller-Charles word pairs, they give comparable results to other methods.

The rest of the paper is structured as follows. We first give a short overview of the different kinds of existing methods used for computing semantic similarity in Section 2. Then, in Section 3, we describe our methods in detail. Finally, in Sections 4 and 5, we demonstrate our results and draw conclusions from them.

2 Related Methods

The methods computing semantic similarity can use a variety of sources and can compute the semantic similarity of words differently. There exist methods that make use of large lexical databases, such as the WordNet or the Roget's Thesaurus. Others issue web search engine queries and process the results. Further, there are also numerous methods that employ large static corpora to extract statistical information in order to solve the problem of semantic similarity. In this section, we would like to give a short overview of all these approaches.

2.1 Methods Using Large Lexical Databases

The methods using large lexical databases access the information stored in these databases and compute word similarity based on the extracted information. Most of them use the WordNet, but others apply the Roget's Thesaurus.

As an example, Jarmasz and Szpakowicz [2] defines the similarity of two words based on their distance in Roget's Thesaurus, i.e. the number of edges between them.

[1] Although WordNet is used for obtaining the lemmas of words, it is not used for anything else. This could be substituted with other methods though.

A bag-of-words method based on WordNet was proposed by Patwardhan and Pedersen [3]. For each input word it creates a feature vector from the words contained in their gloss and the words with distance 1 from that input word. The similarity of words is then defined as the cosine similarity of their vectors.

Tsatsaronis et al. [4] also defines a similarity score using WordNet. To compute this, they consider the distance of words in WordNet, the depth of the nodes between them and the types of relations on the route between them (they use all the relation types that can be found in WordNet). They extended their measure, so that it is able to compare not only words, but also longer texts.

2.2 Web Search-Based Methods

There are also numerous methods that try to estimate the similarity of words by issuing web search queries with the given words and then using the returned page hit counts and snippets. The most important of these are presented next.

Higgins [5] first issues queries with the words to be compared independently. Then, he also issues queries in which the two words are next to each other. Finally, the similarity of these words is defined as their pointwise mutual information computed from the returned page hit counts.

Sahami and Heilman [6] collect the snippets returned for input queries. For each snippet they create a vector with TF-IDF weighting. Then, the vectors are normalized and the centroids of the set of vectors returned for a query are computed. The similarity of queries is computed as the inner product of the centroids associated with them.

The method of Kulkarni and Caragea [7] is composed of two parts. The first part assigns to any input word a set of the most associated words with it, thus creating a concept cloud for each input word. Then, in the second part, these concept clouds are compared, and the semantic similarity of the words is determined by the similarity of their concept clouds. In both parts they issue web search queries.

2.3 Methods Employing Large Static Corpora

Methods in this category usually build a vector for each word based on their contexts found in the used static corpora and define word similarity as the similarity of their vectors. Although our methods are similar to those below, ours use new features, weighting schemes and vector similarity measures in addition to the existing ones.

The method called Latent Semantic Analysis (LSA), introduced by Landauer and Dumais [8], is very similar to Latent Semantic Indexing (LSI). It first creates a matrix of words and chunks of text (e.g. sentences or paragraphs), where the cells contain the weight of the words regarding the chunks of text. Then, it applies Singular Value Decomposition (SVD) to compress this matrix. Finally, it computes the similarity of words based on the similarity of their vectors in this compressed matrix.

The method of Lin [9] assigns a feature set to every input word, which contains those (grammatical relation, feature word)-pairs that co-occurred with the input word in a corpus. Similarity is then defined using the information content of the feature sets of the words as well as the information content of the intersection of their feature sets.

The method proposed by Rapp [10] creates a numerical feature vector for words based on the contexts they have in a corpus. In these vectors those words are contained, that occur within a 2 word window in the used corpus, and their score is based on word association measures such as pointwise mutual information. The matrix formed by these feature vectors is then compressed using SVD. Finally, the similarity of the words is computed as the similarity of their compressed feature vectors.

Gabrilovich and Markovitch [11] maps input texts into weighted vectors over Wikipedia articles, based on the similarity of the texts and the articles. Then, they compute the similarity of input texts as the cosine similarity of their vectors.

2.4 Combined Methods

All methods of the three categories above have their advantages and disadvantages. Methods using lexical databases are usually exact for the words included in them but have poor coverage on specific professional domains. Using large static corpora provides a great opportunity for inspecting the distribution of neighboring words for any word, but they can still provide less than enough data for very rare words. By issuing web search queries there is enough information for almost every word, but such methods have the many disadvantages described in Section 1. Because of these different characteristics, a number of researches tried to combine different types of methods in order to combine the best properties of them, thus creating new methods.

Resnik [12] assigns a probability for each synset in WordNet, based on the probability of the occurrence of its words in a static corpus. The similarity of words is then given as the maximum information content of their least common subsumer synset.

Lin [9] defined another measure, which is very similar to Resnik's [12]. The difference is that beside the information content of the least common subsumer of the input words, it employs the information content of the words' synsets too.

The method of Turney et al. [13] is a combination of 4 different methods. One is the LSA [8], the second is a web search based method called PMI-IR, the third searches in an online thesaurus (Wordsmyth thesaurus online) and the last processes the snippets returned by web queries. These 4 methods were then combined in different ways, such as with the product rule, to provide a final similarity measure.

The hyperlink structure of the Wikipedia was utilized by Milne and Witten [14]. They employ the anchors found in the articles (links to other articles), and they represent each article with the set of its incoming and outgoing links. First, they identify candidate articles for each input word using the text of anchors in the article. Then, after resolving ambiguities using different approaches, they compute the similarity of the articles associated with the input words using the page hit counts of web queries.

Agirre et al. [15] use a combination of two methods. The first assigns a vector to each word by running the PageRank algorithm on WordNet. The other uses statistical co-occurrence data from a corpus of 1.6 Terawords, and has 3 versions: a bag-of-words approach, a version using a static context window and another using dependency relations. In both methods, word similarity is computed using a vector similarity measure. To combine the two main methods, they train an SVM.

3 Our Methods

The main idea behind our approach, as behind most other ones, is that semantically similar words behave similarly and occur in similar contexts. Therefore, our algorithms first create a feature vector for each word based on statistical co-occurrence information gathered from corpora, which is followed by the comparison of word pairs using a vector similarity measurement. There are several variations of our method, using different features, vector types, weights and vector comparison measures. An earlier version of some of these methods was previously described in Dobó [16].

3.1 Feature Extraction

For the task of extracting features from the used corpora we applied two main approaches. The first and simpler one is the bag-of-words approach. It finds each occurrence of the selected word in the used corpora, then includes every word in a window of 3 words within that occurrence in the feature vector. However, it is different from regular bag-of-words approaches, since (in case of using numerical feature vectors) in addition to the weighting scheme it uses, described in Section 3.3, it counts the occurrences of close words multiple times. Specifically, the frequency this method assigns to a feature word is based on the distance of the observed word and the feature word. Several different techniques were tested, the best was found to be using frequencies that scale quadratically with the distance (with a window size of 3, frequency 9 is assigned to distance 1, frequency 4 to distance 2 and frequency 1 to distance 3).

Our other approach uses features based on grammatical relations obtained from the used corpora. Grammatical relations were extracted using the C&C CCG parser [17]. For each word, (grammatical relation, feature word)-pairs are included in the feature vector, where the feature words are those that are in a grammatical relation with the original word, similarly as in Lin [9]. Some example features are (subject-of, word), (object-of, word) and (preposition, word) among others. It is important to note that in this approach paraphrases, prepositions, patientive ambitransitive verbs and passive verbs were treated similarly as in Dobó [16] and in Dobó and Pulman [18].

Both approaches were tested using three corpora, namely the British National Corpus (BNC), the Web 1T 5-gram Corpus (only the 4 and 5-grams), and the corpus of the English Wikipedia[2]. Since any corpus can be used to create the feature vectors, our methods can be easily adapted to different domains and languages if needed.

3.2 Creating and Comparing the Feature Vectors

For creating and comparing the feature vectors of words two main approaches were tested. First, the approach presented by Lin [9] (the one using static corpora and not using WordNet, presented in Section 2.3) was re-implemented with some modifications. This method uses binary feature vectors (i.e. feature sets), indicating whether a

[2] Pre-processed using the wikipedia2text_rsm_mods toolkit by Rafael Mudge, available from http://blog.afterthedeadline.com/2009/12/04/generating-a-plain-text-corpus-from-wikipedia.

feature has occurred with the given word, without weight. Then, to compute the similarity of words, it compares these feature vectors using the similarity measure of Lin [9], which assigns a similarity score of 0 to 1 (inclusive) for a word pair. Although the base of this approach is the same as described in Lin [9], in order to improve its performance there were some changes made in its implementation regarding the treatment of paraphrases, patientive ambitransitive verbs and passive verbs, similarly as in Dobó [16] and in Dobó and Pulman [18].

The previous approach does not take into account the frequency with which a feature co-occurred with a word. But, this co-occurrence frequency also contains useful information, so it is logical to try to use that information too. Therefore another method was created that does not only store the features for a word in a set, rather it creates a weighted numerical vector from them. Then, the similarity of these vectors provides the similarity of words. The types of weights and the vector comparison measures used in this approach are described in Section 3.3 and 3.4, respectively.

3.3 Weighting Inside the Numerical Feature Vectors

Weighting can be used to assign importance to the features, and thus to consider different aspects of the features significant. Seven different weighting schemes were tested inside the numerical feature vectors. The description of these follows now.

The simplest of them is the co-occurrence frequency of the (word, feature) pairs (freq). In this case, the importance of a feature is based on the number of times it co-occurred with the word.

The second weighting is a slightly modified version of the first one. Instead of the simple frequencies of the (word, feature) pairs, the logarithm of this frequency is stored, with a smoothing parameter of 1 (logfreq). This way, (word, feature) pairs with very high frequency cannot be overweighted.

The problem of the first two weighting methods is that they assign an overly high importance to those features that occur very frequently in any context, such as the features (subject-of, be) or (object-of, have). It would be better to assign a low importance to features like these, since they do not tell much information about the words they are connected to, and assign high importance to those features instead that are specific to the words. One such measure is the pointwise mutual information (pmi) [19], which measures association strength. This was chosen as the third type of weight. However, since it is unstable for very small counts [19], (word, feature) pairs with a frequency of at most 5 are discarded when using this weighting scheme.

Another way for testing the strength of association is using the log-likelihood ratio [20], which was also employed as a weight in the numerical feature vectors (loglh).

The fifth measure was a combination of two different measures. The first is the logarithm of the frequency of the (word, feature) pairs. But the problem is, as noted before, that it assigns a high value for the most common features that are not specific to any word. This is compensated by the second part, which is the logarithm of the number of words that occur with the given feature. Both logarithms use a smoothing parameter of 1. The combined measure is calculated as the quotient of the two parts:

$$qw(x,y) = \frac{\ln(1+c_{xy})}{\ln(1+f(y))} \tag{1}$$

where c_{xy} is the frequency of the co-occurrence of x and y, and $f(y)$ is the number of words, with which feature y occurs [9].

The sixth weight is also a combined measure. Its first part is again the logarithm of the frequency of the (word, feature) pairs. And its second part, which is the logarithm of the information content of the feature, is again used for compensation. In case of both logarithms, a smoothing parameter of 1 was used. The two parts are multiplied together to form the combined measure:

$$pw(x,y) = \ln(1+c_{xy}) \times \ln(1+I(y)) \tag{2}$$

where c_{xy} is same as before and $I(y)$ is the information content of feature y [9].

The last implemented weighting measure is the entropy-based measured used by Rapp [10] (rapp).

3.4 Similarity Measures for the Numerical Feature Vectors

Two frequently used vector similarity measures were tested in the algorithms with numerical feature vectors. The first was the cosine similarity, which compares two vectors by computing the cosine of the angle between them [20]. The second was a generalization of the Dice coefficient. In its original form, it can only compute the similarity between Boolean vectors. In order to use it for numerical vectors, we used its generalization proposed by Lin [9]. Both similarity measures return a similarity value between -1 and +1 (inclusive).

3.5 Determining the Part-of-Speech of the Input Words

There are many words that can take more than one part-of-speech (POS). For example, the words run and bank can be both nouns and verbs. When these types of words are used with different POSs, different features are relevant. Therefore, first the POS of the input words needs to be determined, and the feature vectors can only be created after this. The POS of these controversial words can be inferred from the other words contained in the same question. For our methods, we assumed that each input word is a verb, noun, adjective or adverb and each question contains words of the same POS.

For a question the part-of-speech maximizing the following formula is chosen:

$$pos = \arg\max_{p} \prod_{w \in q} \ln(1.0001 + f_{w,p}) \tag{3}$$

where p can take any of the four possible POSs, q denotes the question, w runs through the words of q and $f_{w,p}$ is the frequency of w having p part-of-speech.

Table 1. Results on the Miller-Charles dataset (Spearman correlation). Notations: bnc/enwiki /web1t5gram denotes the corpus; bagofwords/parsed denotes the used feature types (bag-of-words or grammatical relations); lin/num denotes the method (the one based on Lin [9] or the one using numerical feature vectors); cos/dice denotes the similarity measure; freq/logfreq/pmi/ loglh/qw/pw/rapp denotes the weighting scheme; + denotes the combination of two methods.

Method	**Result**
bnc-bagofwords-num-cos-qw+enwiki-parsed-num-cos-freq	0.773
bnc-bagofwords-num-cos-qw+enwiki-parsed-num-cos-qw	0.750
enwiki-bagofwords-num-cos-pmi+bnc-parsed-num-cos-qw	0.737
enwiki-bagofwords-num-cos-pmi+enwiki-parsed-num-cos-pmi	0.729
enwiki-parsed-num-cos-pmi	0.727
bnc-parsed-num-cos-loglh+enwiki-parsed-num-cos-pmi	0.712
enwiki-bagofwords-num-cos-pmi	0.684
enwiki-parsed-num-dice-pmi	0.661
web1t5gram-parsed-num-cos-loglh	0.631
enwiki-bagofwords-num-cos-pmi+enwiki-parsed-lin	0.616
bnc-parsed-lin	0.417

3.6 Combination of the Individual Methods

In order to combine the strengths of the different methods and achieve better results, not only the above described methods, but their combinations were also tested. When two methods were combined, the similarity score for each word pair was calculated separately. Afterwards, the logarithm of the scores (with a smoothing parameter of 1) were multiplied together to form the similarity score of the combined method. Taking the logarithm of the scores before multiplying them helps balancing the results: we consider a word pair having two moderate scores better than a word pair having a very low and a very high score.

4 Results

All the methods described in the previous section were tested on two different datasets, namely the Miller-Charles word pairs (MC) and the TOEFL synonym questions. Both data sets were widely used in the evaluation of methods computing semantic similarity by others. The first one contains 30 word pairs, for which a similarity score between 0 to 4 was assigned by 38 undergraduate students. Since there were words in 2 word pairs that were not included in previous WordNet versions, in most research these pairs were omitted. Consequently, only the remaining 28 word pairs were used here as well. The other data set contains 80 synonym questions from the TOEFL language exam. In all of the questions, a question word is given with 4 alternatives, and the task is to determine the most similar word to the question word.

Table 2. Results on the TOEFL questions (percent of correct answers)

Method	**Result**
bnc-parsed-num-cos-loglh+enwiki-parsed-num-cos-pmi	88.75%
enwiki-bagofwords-num-cos-pmi+enwiki-parsed-num-cos-pmi	87.50%
enwiki-bagofwords-num-cos-pmi+bnc-parsed-num-cos-qw	86.25%
enwiki-bagofwords-num-cos-pmi	83.75%
enwiki-parsed-num-cos-pmi	82.50%
enwiki-bagofwords-num-cos-pmi+enwiki-parsed-lin	81.25%
enwiki-parsed-num-dice-pmi	78,75%
bnc-bagofwords-num-cos-qw+enwiki-parsed-num-cos-qw	77.50%
bnc-bagofwords-num-cos-qw+enwiki-parsed-num-cos-freq	72.50%
bnc-parsed-lin	68.75%
web1t5gram-parsed-num-cos-loglh	60.00%

Table 3. Comparison with other results on the Miller-Charles dataset (Spearman correlation)

Method	**Result**	**Used data**
Human upper bound [12]	0.934	
Agirre et al. [15]	0.92	WordNet, corpus
Patwardhan and Pedersen [3]	0.91	WordNet
Jarmasz and Szpakowicz [2]	0.87	Roget's Thesaurus
Tsatsaronis et al. [4]	0.856	WordNet
Kulkarni and Caragea [7]	0.835	Web search
Lin [9]	0.82	WordNet, corpus
Resnik [12]	0.81	WordNet, corpus
bnc-bagofwords-num-cos-qw+ enwiki-parsed-num-cos-freq	0.773	corpus
bnc-bagofwords-num-cos-qw+ enwiki-parsed-num-cos-qw	0.750	corpus
enwiki-bagofwords-num-cos-pmi+ bnc-parsed-num-cos-qw	0.737	corpus
enwiki-bagofwords-num-cos-pmi+ enwiki-parsed-num-cos-pmi	0.729	corpus
Gabrilovich and Markovitch [11]	0.72	corpus
bnc-parsed-num-cos-loglh+ enwiki-parsed-num-cos-pmi	0.712	corpus
Milne and Witten [14]	0.70	Wikipedia links, Web search
Sahami and Heilman [6]	0.618	Web search

In case of the MC dataset, the average scores of the 38 students were used, and the evaluation was done by computing the Spearman correlation of these scores and the scores returned by our methods. When testing with the TOEFL questions, the evaluation measure was the percentage of the correct answers given.

The results of some selected methods are presented in Tables 1 and 2. It can be seen that the best performance was 0.773 on the MC dataset and 88.75% on the TOEFL questions, both achieved by a combined method. The best results of individual methods (without combination) were 0.727 and 83.75%, respectively. The scores of the methods using numerical feature vectors were mostly higher than the scores of the approach based on Lin [9], and still better results were achieved by combining the different methods. The methods that achieved best performance considering both datasets were the *bnc-parsed-num-cos-loglh+enwiki-parsed-num-cos-pmi* (MC: 0.712, TOEFL: 88.75%), the *enwiki-bagofwords-num-cos-pmi+enwiki-parsed-num-cos-pmi* (MC: 0.729, TOEFL: 87.50%) and the *enwiki-bagofwords-num-cos-pmi+bnc-parsed-num-cos-qw* (MC: 0.737, TOEFL: 86.25%). Comparison with other methods is shown in Tables 3 and 4, which shows that our methods had an average performance on the MC dataset, while one of our methods achieving the 3rd best score on the TOEFL questions. Most importantly, however, if we only take those methods into account that solely use static corpora, our methods perform 1st and 2nd best on the two datasets, respectively.

5 Conclusion and Future Work

In this article we have demonstrated methods computing the semantic similarity of words that compare favorably to other measures. They use statistical co-occurrence data extracted from static corpora to create feature vectors for the input words, and then define the similarity of the words as the similarity of their vectors. Several variations were created, using different features, vector types, weights and vector comparison measures, and combinations of our individual methods were also examined.

All these methods were tested on two datasets, namely the Miller-Charles dataset (MC) and the TOEFL synonym questions. On the MC dataset our best method had an average performance (0.773), with many others achieving better results. On the other hand, our best accuracy of 88.75% on the TOEFL questions is 3rd best overall and is much higher than the score achieved by an average non-English US college applicant. When comparing our methods with only those methods that solely rely on static corpora, according to our best knowledge they reach 1st and 2nd place on the two datasets, respectively.

Based on that, we think that our best methods could be successfully used for solving real-life problems too. The fact that they perform better on the TOEFL questions than on the MC dataset indicates that they are more suitable for selecting the most similar word for an input word from a list of candidates than giving an exact similarity value for a pair of words.

In the future, it would be worthy to test our methods with even larger corpora, as more data can result in better accuracy (for example, Agirre et al. [15] use a corpus of

1.6 Terawords and run their algorithm on 2000 CPU cores). As any corpus can be used to extract co-occurrence information, our methods could easily be adapted to different languages (especially in case of the bag-of-words approach). Therefore, we would like to try our algorithms with languages other than English, too. Furthermore, as described in Section 2.4, methods that are a combination of different types of methods can combine the advantages of those methods combined. We therefore think that by creating a combined method whose one method is ours and the other method(s) is (are) using web search engines or large lexical databases, our results could be further improved.

Table 4. Comparison with other results on the TOEFL questions (percent of correct answers)

Method	**Result**	**Used data**
Turney et al. [13]	97.5%	Web search, thesaurus
Rapp [10]	92.5%	corpus
bnc-parsed-num-cos-loglh+ enwiki-parsed-num-cos-pmi	88.75%	corpus
enwiki-bagofwords-num-cos-pmi+ enwiki-parsed-num-cos-pmi	87.50%	corpus
Tsatsaronis et al. [4]	87.5%	WordNet
enwiki-bagofwords-num-cos-pmi+ bnc-parsed-num-cos-qw	86.25%	corpus
enwiki-bagofwords-num-cos-pmi	83.75%	corpus
Higgins [5]	81.3%	Web search
Jarmasz and Szpakowicz [2]	78.7%	Roget's Thesaurus
Average non-English US college applicant [8]	64.5%	
Landauer and Dumais [8]	64.3%	corpus
Lin [9]	24.0%	WordNet, corpus
Resnik [12]	20.3%	WordNet, corpus

References

1. Kilgarriff, A.: Googleology is bad science. Computational Linguistics. 33, 147–151 (2007).
2. Jarmasz, M., Szpakowicz, S.: Roget’s Thesaurus and Semantic Similarity. In: 4th Conference on Recent Advances in Natural Language Processing. pp. 212–219. John Benjamins Publishers, Amsterdam (2003).
3. Patwardhan, S., Pedersen, T.: Using WordNet-based Context Vectors to Estimate the Semantic Relatedness of Concepts. In: 11th Conference of the European Chapter of the Association for Computational Linguistics. pp. 1–8. Association for Computational Linguistics, Stroudsburg (2006).

4. Tsatsaronis, G., Varlamis, I., Vazirgiannis, M.: Text Relatedness Based on a Word Thesaurus. Journal of Artificial Intelligence Research. 37, 1–39 (2010).
5. Higgins, D.: Which Statistics Reflect Semantics? Rethinking Synonymy and Word Similarity. In: Kepser, S. and Reis, M. (eds.) Linguistic Evidence: Empirical, Theoretical and Computational Perspectives. pp. 265–284. Mouton de Gruyter, Berlin, New York (2005).
6. Sahami, M., Heilman, T.D.: A web-based kernel function for measuring the similarity of short text snippets. In: 15th international conference on World Wide Web. pp. 377–386. ACM Press, New York (2006).
7. Kulkarni, S., Caragea, D.: Computation of the Semantic Relatedness between Words using Concept Clouds. In: International Conference on Knowledge Discovery and Information Retrieval. pp. 183–188. INSTICC Press, Setubal (2009).
8. Landauer, T.K., Dumais, S.T.: A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. Psychological Review. 104, 211–240 (1997).
9. Lin, D.: An information-theoretic definition of similarity. In: 15th International Conference on Machine Learning. pp. 296–304. Morgan Kaufmann Publishers Inc., San Francisco (1998).
10. Rapp, R.: Word Sense Discovery Based on Sense Descriptor Dissimilarity. In: 9th Machine Translation Summit. pp. 315–322. Association for Machine Translation in the Americas, Stroudsburg (2003).
11. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In: 20th International Joint Conference on Artificial Intelligence. pp. 1606–1611. Morgan Kaufmann Publishers Inc., San Francisco (2007).
12. Resnik, P.: Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In: 14th International Joint Conference on Artificial Intelligence. pp. 448–453. Morgan Kaufmann Publishers Inc., San Francisco (1995).
13. Turney, P.D., Littman, M.L., Bigham, J., Shnayder, V.: Combining Independent Modules to Solve Multiple-choice Synonym and Analogy Problems. In: 4th Conference on Recent Advances in Natural Language Processing. pp. 482–489. John Benjamins Publishers, Amsterdam (2003).
14. Milne, D., Witten, I.H.: An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links. In: 23rd AAAI Conference on Artificial Intelligence. pp. 25–30. AAAI Press, Menlo Park (2008).
15. Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., Soroa, A.: A study on similarity and relatedness using distributional and WordNet-based approaches. In: 10th Annual Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies. pp. 19–27. Association for Computational Linguistics, Stroudsburg (2009).
16. Dobó, A.: Angol szavak szinonimáinak automatikus keresése. National Scientific Conference of Students (OTDK). OTDT, Budapest (2011).
17. Clark, S., Curran, J.R.: Parsing the WSJ using CCG and log-linear models. In: 42nd Annual Meeting on Association for Computational Linguistics. pp. 103–110. Association for Computational Linguistics, Stroudsburg (2004).
18. Dobó, A., Pulman, S.G.: Interpreting noun compounds using paraphrases. Procesamiento del Lenguaje Natural. 46, 59–66 (2011).
19. Church, K.W., Hanks, P.: Word association norms, mutual information, and lexicography. Computational Linguistics. 16, 22–29 (1989).
20. Manning, C., Schütze, H.: Foundations of statistical natural language processing. MIT Press, Cambridge (2000).