Unsupervised WSD with a Dynamic Thesaurus*

Javier Tejada-Cárcamo,[1,2] Hiram Calvo[1], Alexander Gelbukh[1]

[1] Center for Computing Research, National Polytechnic Institute, Mexico City, 07738, Mexico
[2] Sociedad Peruana de Computación, Arequipa, Peru
jawitejada@hotmail.com, hcalvo@cic.ipn.mx, gelbukh@gelbukh.com

Abstract. Diana McCarthy *et al*. (ACL-2004) obtain the predominant sense for an ambiguous word based on a weighted thesaurus of words related to the ambiguous word. This thesaurus is obtained using Dekang Lin's (COLING-ACL-1998) distributional similarity method. Lin averages the distributional similarity by the whole training corpus; thus the list of words related to a given word in his thesaurus is given for a word as type and not as token, i.e., does not depend on a context in which the word occurred. We observed that constructing a list similar to Lin's thesaurus but for a specific context converts the method by McCarthy *et al*. into a word sense disambiguation method. With this new method, we obtained a precision of 69.86%, which is even 7% higher than the supervised baseline.

1 Introduction

Word Sense Disambiguation (WSD) task consists in determining the intended sense of an ambiguous word in a specific context. For example, *doctor* has three senses listed in WordNet: (1) *person who practices medicine*, (2) *person who holds Ph.D. degree from an academic institution*; and (3) *a title conferred on 33 saints who distinguished themselves through the orthodoxy of their theological teaching*. The WSD task consists in determining which sense is intended, e.g., in the context *The doctor prescribed me a new medicine*. This task is important, for example, in information retrieval, where the user expects the documents be selected based on a particular sense of the query word; in machine translation and multilingual querying systems, where an appropriate translation of the word must be chosen in order to produce the translation or retrieve the correct set of documents, etc.

The WSD task is usually addressed in two ways: (1) supervised learning: applying machine-learning techniques trained on previously hand-tagged documents and (2) unsupervised learning: automatically learning, directly from raw word grouping, clues that lead to a specific sense, according to the hypothesis that different words have similar meanings if they occur in similar contexts [4, 6].

The Senseval competitions are devoted to the advances of the state-of-the-art methods for WSD. For instance, the results of Senseval-2 English all-words task are presented in Table 1. This task consists of 5,000 words of running text from three

* Work done under partial support of Mexican Government (CONACyT, SNI) and IPN (PIFI, SIP, COTEPABE). The authors thank Rada Mihalcea for useful comments and discussion.

Penn Treebank and Wall Street Journal articles. The total number of words that are to be disambiguated is 2,473. Sense tags are assigned using WordNet 1.7. The third column in the table shows whether a particular system uses manually tagged data for learning. As one can notice, the best systems are those which learn from previously manually tagged data. However, such a resource is not available for every language, and it can be costly to build. Because of this, we will focus on unsupervised methods, such as those used by the system UNED-AW-U2.

Table 1. Top-10 Systems of Senseval-2

Rank	System	Type	Precision	Recall	Attempted
1	SMUaw	supervised	0.690	0.690	100%
2	CNTS-Antwerp	supervised	0.636	0.636	100%
3	Sinequa-LIA - HMM	supervised	0.618	0.618	100%
–	*WordNet most frequent sense*	*supervised*	0.605	0.605	100%
4	UNED - AW-U2	unsupervised	0.575	0.569	98.908
5	UNED - AW-U	unsupervised	0.556	0.550	98.908
6	UCLA - gchao2	supervised	0.475	0.454	95.552
7	UCLA - gchao3	supervised	0.474	0.453	95.552
8	CL Research - DIMAP	unsupervised	0.416	0.451	
9	CL Research - DIMAP (R)	unsupervised	0.451	0.451	100%
10	UCLA - gchao	supervised	0.500	0.449	89.729

Choosing always the most frequent sense for each word yields a precision and recall of 0.605. The most frequent sense heuristic is a good strategy, since the baseline of 60% would be ranked among the first four systems; we included this algorithm in Table 1 for comparison. The most frequent sense can be obtained from WordNet: it is listed there first in the list of senses for a word—more specifically, the senses in WordNet are ordered according to the frequency data in a manually tagged corpus SemCor [10]; senses that do not occurr in SemCor are ordered arbitrarily. Therefore, any algorithm relying on WordNet ordering of senses is supervised; in particular, it is not applicable for languages or genres for which the frequency data is not available.

McCarthy *et al.* [6] proposed an unsupervised algorithm to find the predominant sense for each word, without addressing the WordNet/SemCor frequency data. They rely on the Lin thesaurus [4] for determine word relatedness. Given a word w, they consider all words u related to w in the Lin thesaurus. They choose a sense ws_i of w and a sense us_j of u that maximize a sense relatedness measure between senses in WordNet. The word u is then said to *vote* for the sense ws_i of w; the *strength* of such a vote is a combination of the sense relatedness between ws_i and us_j in WordNet and the word relatedness between w and u in the Lin thesaurus. The sense ws_k that receives more and stronger votes is predicted to be the predominant sense of the word w and, in particular, can be used in the most frequent sense heuristic for WSD; see Figure 1.

Note that this WSD method does not use at all the context of the word; it always assigns the same sense to the same string regardless of the context. The sense chosen

as predominant for a word depends solely on the corpus used to build the thesaurus, i.e., this information is tied to a word as type and not as token.

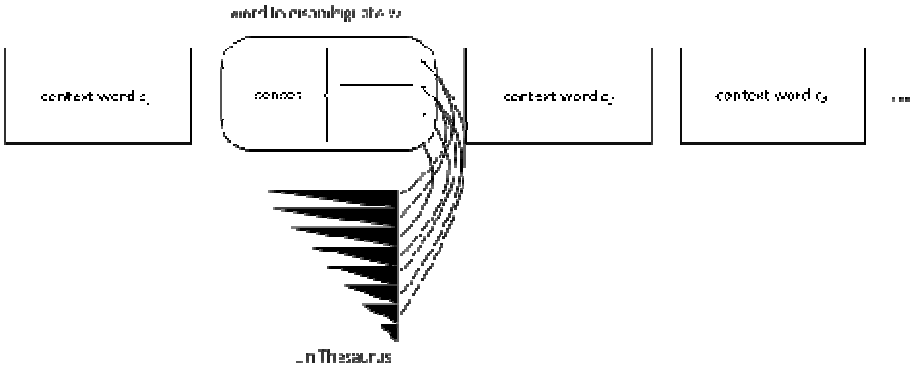


Figure 1. Finding the predominant sense using a static thesaurus as in [6].

In this paper we propose considering context words to dynamically build a thesaurus of words related to a specific occurrence (token) of the word to be disambiguated. This thesaurus is built based on a dependency co-occurrence database (DCODB) previously collected from a corpus. Each *co-occurrent-with-context* word votes for a sense of the word in question as in the McCarthy *et al.*'s method, but in this case this gives the most suitable sense for this word *in a particular context*; see Figure 2.

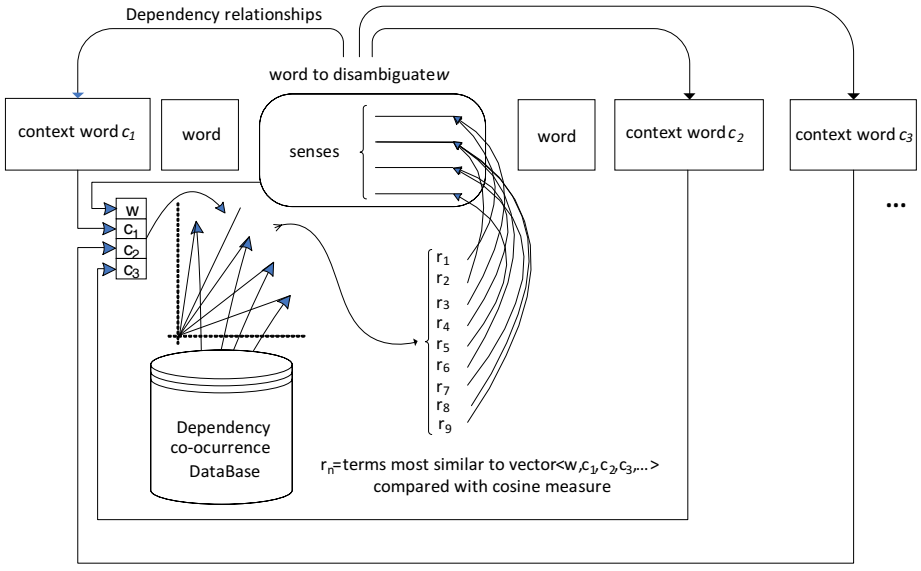


Figure 2. Our proposal: create a dynamic thesaurus based on the dependency context of the ambiguous word.

In Section 2.1 below we explain how the Dependency Co-Ocurrence DataBase (DCODB) resource is built. In Section 2.2 we explain our way of measuring the relevance of co-occurrences based on information theory. In Sections 2.3 and 2.4, we explain more details of our method. In Section 3 we present experimental results that show that the performance of our method is as good as that of some supervised methods. Finally, in Section 4 we draw the conclusions.

2 Methodology

2.1 Building the Dependency Co-Occurrence Database (DCODB)

Dependency relationships are asymmetric binary relationships between a *head* word and a *modifier* word. A sentence builds up a tree which connects all words in it. Each word can have several modifiers, but each modifier can modify only one word [1, 7].

We obtain dependency relationships in a given corpus automatically using the MINIPAR parser. MINIPAR has been evaluated with the SUSANNE corpus, a subset of the Brown Corpus, where it recognized 88% of the dependency relationships with an accuracy of 80% [5]. We apply three simple heuristics for extracting *head-governor* pairs of dependencies:

1. Ignore prepositions; see Figure 3.
2. Include sub-modifiers as modifiers of the head; see Figure 4.
3. Separate heads that are lexically identical but have different parts of speech; this helps to keep contexts separated.

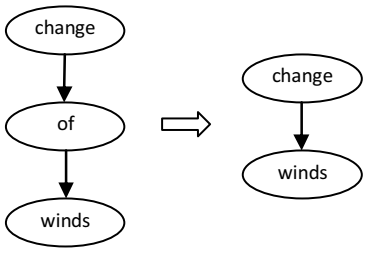


Figure 3. Ignoring prepositions.

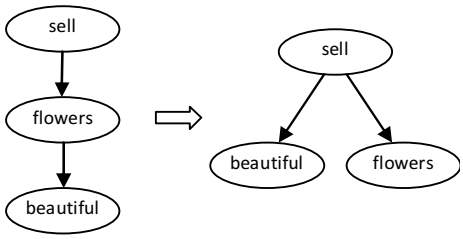


Figure 4. Including sub-modifiers as modifiers of the head.

2.2 Statistical Model

We use the vector-space model with TF-IDF (term frecuency—inverse document frequency) weighting. This model is often used for classification tasks and for measuring document similarity. Each document is represented by an n-dimensional vector, where n is the number of different words (types) in all documents of the collection. In our method, we treat a *head* as a document title, and all its modifiers in the

corpus as the contents of such a "documents" that gives a vector corresponding to this head. The TF (term frequency) value for each dimension of this vector is the number of times this modifier modified this head in the training corpus (normalized as explained below). We can represent the vector as

$$Vector(head_j) = \{(mod_{1,j}, f_{1,j}), (mod_{2,j}, f_{2,j}), ..., (mod_{n,j}, f_{n,j})\},$$

where:

$head_j$ is the given head word,
$mod_{i,j}$ is a modifier word,
$f_{i,j}$ is the a normalized number of times $mod_{i,j}$ modified $head_j$:

$$f_{i,j} = \frac{freq_{i,j}}{\max(freq_{l,j})},$$

where:

$freq_{i,j}$ is the frequency of the modifier i with $head_j$,
$\max(freq_{i,j})$ is the highest frequency number of the modifiers of $head_j$.

The weight w_{ij} of the modifier i for the head j is a product of the normalized frequency vector $\{f_{i,j}\}$ of the head (TF) and its inverse frequency (IDF). TF shows the importance of each modifier with respect to the modified head, so that the weight of their relationship increases when the modifier appears more frequently with this head. IDF shows the relevance of a modifier with regard to the other heads in the database (DCODB), in such a way that the weight of the modifier decreases if it appears more often with other heads in the DCODB, and increases when it appears with fewer heads. This means that very frequent modifiers do not help to discriminate heads. IDF is calculated as

$$idf_i = \log \frac{N}{n_i},$$

where:

N is the total number of heads,
n_i is the total number of heads which are at least once modified by modifier i.

Given a training corpus, building the database of the vectors associated with all heads as explained above is a one-time process.

2.3 Disambiguation Process

Given the database described above, we can disambiguate a given word w in a context C made up of words: $C = \{c_1, c_2, \ldots c_n\}$. The first step for this consists in obtaining a weighted list of terms related with w. The second step consist in using these terms to choose a sense of w as in the algorithm by McCarthy *et al.* [6]. The following sections explain these steps in detail.

2.3.1 Obtaining the Weighted List of Terms Related with *w*

A word is related with another one if they are used in similar contexts. In our method this context is defined by syntactic dependencies; see Figure 2. Given an ambiguous word *w*, its dependencies c_1, c_2, c_3, etc. form a vector $\vec{w}$ = <*w*, c_1, c_2, c_3, …, w_j, …>. We compare it with all vectors $\vec{r_i}$ = <$r_{i,1}, r_{i,2}, r_{i,2}, \ldots r_{i,j}, \ldots$> from DCODB using the cosine similarity measure:

$$cos\_measure(\vec{w}, \vec{r_i}) = \frac{\vec{w} \cdot \vec{r_i}}{\left|\vec{w}\right| \times \left|\vec{r_i}\right|} = \frac{\sum_{j=1}^{n} w_j \times r_{i,j}}{\sqrt{\sum_{j=1}^{n} (w_j)^2} \times \sqrt{\sum_{j=1}^{n} (r_{i,j})^2}}.$$

The value obtained is used as a similarity weight for creating the weighted list of related terms. Note that this procedure suffers the data sparseness problem, because the number of modifiers of an ambiguous word is between 0 and 5—considering only one level of the syntactic tree—whereas the number of non-zero coordinates in the majority of vectors in the DCODB is much higher. Table 2 shows an example of calculation of the cosine measure. Given the vector $\vec{w}$ formed by the word *w* and its context words (based on dependency relationships from the sentence where *w* is found), the DCODB is queried with all the r_n words to compare with each vector $\vec{r_n}$. For example, the cosine measure between $\vec{w}$ and $\vec{r_3}$ is given by:

$$cos\_measure(\vec{w}, \vec{r_3}) = \frac{(1\cdot4)+(1\cdot3)+(1\cdot1)+(1\cdot0)+(0\cdot0)+(0\cdot0)+(0\cdot4)+(0\cdot4)}{\sqrt{1^2+1^2+1^2+1^2+0^2+0^2+0^2+0^2}+\sqrt{4^2+3^2+1^2+0^2+0^2+0^2+4^2+4^2}} = 0.832.$$

Table 2. Example of cosine measure calculation

	← j →										
	c_1	c_2	c_3	…	c_n	o_1	o_2	o_3	…	o_m	*cos_measure*
w	**1**	**1**	**1**	**…**	**1**	**0**	**0**	**0**		**0**	1
r_1	1	6	2	…	0	0	0	3		0	0.992165
r_2	0	4	1	…	3	4	1	1		0	0.92665
r_3	4	3	1	…	0	0	0	4		4	0.831966
…										…	
r_{13}	0	0	2	…	4	0	0	1		5	0.68319
…										…	

2.3.2 Voting Algorithm

Here we describe our modifications to the voting algorithm by McCarthy *et al.* [6]. This algorithm allows each member of the list of related terms (thesaurus that is dynamic in our proposal or static in [6]) to contribute for a particular sense of the ambiguous word *w*. The weight of the term in the list is multiplied by the semantic distance between each of the senses of a term $r_i s_j$ and the senses of the ambiguous word ws_k. The highest value of semantic distance determines the sense of *w* for which the term r_i votes. Once all terms r_i have voted (or a limit has been reached), the sense of *w* which

```
Sense voting algorithm:
    for each ambiguous word w
        build vector w⃗ =<w,c1,c2,...,cn> of its context words
        for each vector r⃗i in DCODB,
            calculate weight(r⃗i)=cos_measure(w⃗,r⃗i)
            sort vectors r⃗ from highest to lowest cos_measure; r⃗1 has the greatest weight
            for each word ri corresponding to the head of each vector r⃗i
                for each sense risj of word ri
                    for each sense wsk of word w,
                        calculate a=max(a,similarity(risj, wsk)·weight(r⃗i))
                    sense wsk corresponding to the last maximum receives a vote of a units.
                    stop if i > max_neighbors (maximum number of vectors)
    return max(votes(wsk))
```

Figure 5. Sense voting algorithm

received more and stronger votes is selected. See Figure 5 for the pseudo-code of the algorithm.

In the following section we describe the measure of similarity used in this algorithm.

2.4 Similarity Measure

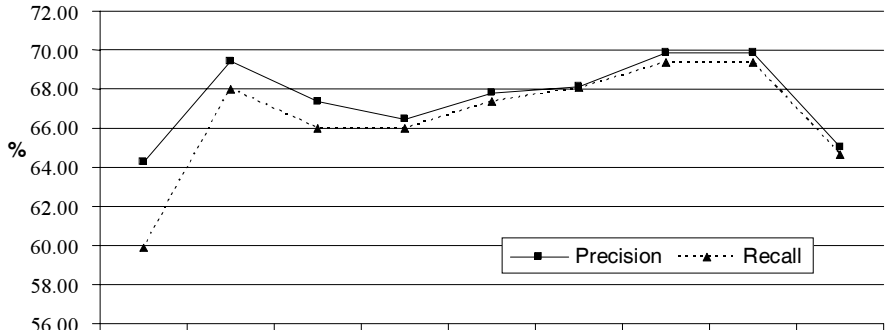
To calculate the semantic distance between two senses we use WordNet::Similarity package [11]. This package is a set of libraries which implement similarity measures and semantic relationships in WordNet [8, 9]. It includes similarity measures proposed by Resnik [12], Lin [4], Jiang-Conrath [2], Leacock-Chodorow [3], among others. In order to follow McCarthy *et al.* approach, we have chosen the Jiang-Conrath similarity measure as they did. The Jiang-Conrath measure (jcn) uses exclusively the hyperonym and hyponym relationships in the WordNet hierarchy, which is consistent with our tests because we consider only disambiguation of nouns. The Jiang-Conrath measure obtained the second best result in the experiments presented by Pedersen *et al.* [10]. In that work they evaluate several semantic measures using the WordNet::Similarity package. The best result was obtained with the adapted Lesk measure [10], which uses information of multiple hierarchies but is less efficient.

The Jiang-Conrath measure uses a concept of information content (IC) to measure the specificity of a concept. A concept with a high IC is very specific—for example, *dessert_spoon*—while a concept with a lower IC is more general, such as *human_being*. WordNet::Similarity uses SemCor to compute the IC of WordNet concepts. The Jiang-Conrath measure is defined as follows:

$$dist_{jcn}(c_1, c_2) = \mathrm{IC}(c_1) + \mathrm{IC}(c_2) - 2 \times \mathrm{IC}(lcs(c_1, c_2)),$$

where:

IC is the information content,
lcs (*lowest common subsumer*) is the lowest common node of two concepts.



	10	20	30	40	50	60	70	100	1000
Precision	64.23	69.44	67.36	66.43	67.80	68.15	69.86	69.86	65.06
Recall	59.86	68.02	65.98	65.98	67.34	68.08	69.38	69.38	64.62

Maximum number of neighbors

Figure 6. Automatic tagging of 10% SemCor, varyinig *max_neighbors*.

3 Experimental Results

We tested our approach on English text. The dependency co-occurrence database (DCODB) was built from raw data (i.e., without considering the sense tags; thus our method is unsupervised) taken from 90% of SemCor. Then, we evaluated the approach by automatically tagging the remaining 10% of SemCor and comparing the results with the hand-tagged senses. We have experimented with different values of *max_neighbors*—the number of most similar vectors from the DCODB that we consider for disambiguating each word. We have tested the top-10, 20, 30, 40, 50, 70, 100, and 1000 most similar terms from our dynamic thesaurus; see Figure 6.

The values of precision and recall are similar because the Jiang-Conrath measure always returns a similarity value when two concepts are compared. Slight deviations are due to inexistence of certain ambiguous words in WordNet. The best results were 69.86% by using 70 neighbors training with 90% of SemCor (used as raw text) evaluated with the remaining 10% as the gold standard.

The most frequent sense of 90% of the manually annotated SemCor corpus against the 10% as gold standard of the same corpus yields a precision and recall of 62.84 (coverage is 100%); so our results are approximately 7% higher than the supervised most frequent sense heuristic.

McCarthy *et al.* report 64% using a term list from a static Lin thesaurus. However, they present results with SENSEVAL-2, so that we cannot make a direct comparison; however, we can do a rough estimation, given that using the most frequent sense heuristic with the frequency counts from SemCor yields 62.84 while using the most frequent sense heuristic with the frequency counts from SENSEVAL-2 corpus yields 60%.

4 Conclusions

The method by McCarthy *et al.* [6] to obtain the predominant sense of a word considers a weighted list of terms related to that word as type (not token). It can be used for WSD task with very good results in the class of unsupervised methods. The related terms are obtained by Lin's method for building a thesaurus [4]. In the WSD task, this list is always the same for each occurrence of a word (token), because it does not depend on the context; this is why we called this method *static thesaurus*. In our method, that list is different for each occurrence of the word (token) depending on both its syntactic (dependency) context and the corpus used for building the co-occurrence database.

Our method is also unsupervised. It disambiguates a corpus more with better precision when trained with (another section of) the same corpus, as shown by our experiments when training with 90% of SemCor used as raw text (this is why our method is unsupervised) and evaluating with the remaining 10%. We compared this against obtaining the most frequent sense from the same subset of SemCor and evaluating with the remaining 10% (which is a supervised method). Our precision was 69.86% vs. 62.84% of that supervised baseline, while our method is unsupervised.

As future work, we plan to carry out more experiments by comparing our results with the hand-tagged section of the SENSEVAL-2 corpus, all-words test. This will allow us to compare our system with the systems presented at of SENSEVAL-2. We believe the results might be similar, given that using the most frequent sense heuristic with frequency counts from SemCor yields a 62.84 while using the frequency counts from SENSEVAL-2 corpus yields 60%.

SemCor is a small corpus, ca. 1 million words, compared with the British National Corpus (BNC, approximately 100 million words). As part of our future work, we plan to train our method with BNC.

On the other hand, Figure 6 is very irregular and we would not say that the optimum number of *max_neighbors* is always 70. In addition, terms from the weighted list (our dynamic thesaurus) are not always clearly related with each other. We expect to build a resource to improve the semantic quality of such terms.

Finally, it is difficult to determine the main factor that has greater impact on the proposed disambiguation method: the process of obtaining a weighted list of terms (the dynamic thesaurus) or the maximization algorithm. This is because the DCODB sometimes does not provide terms related with a word, and besides, the definitions for each sense of WordNet are sometimes very short. In addition, as it has been stated previously, for several tasks the senses provided by WordNet are very fine-graded, so that a semantic measure may be not accurate enough.

References

1. Hays, D. (1964). Dependency theory: a formalism and some observations. *Language*, 40 (4): 511–525.
2. Jiang, J., & Conrath, D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *International Conference on Research in Computational Linguistics*. Taiwan.

3. Leacock, C., M. Chodorow. (1998). Combining local context and WordNet similarity for word sense identification. C. Fellbaum (ed.) *WordNet: An electronic lexical database*, 265–283.
4. Lin, D. (1998). Automatic retrieval and clustering of similar words. *COLING-ACL 98*. Canada.
5. Lin, D. (1998). Dependency-based Evaluation of MINIPAR. *Workshop on the Evaluation of Parsing Systems*. Spain.
6. McCarthy, D., Koeling, R., Weeds, J., Carroll, J. (2004). Finding predominant senses in untagged text. *42nd Annual Meeting of the Association for Computational Linguistics*. Spain.
7. Mel'čuk, I. A. (1987). *Dependency syntax; theory and practice*. Albany, N.Y.: State University of New York Press.
8. Miller, G. (1993). *Introduction to WordNet: An On-line Lexical Database*. Princeton Univesity.
9. Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J. (1990). Introduction to WordNet: An On-line Lexical database. *International Journal of Lexicography*, 3: 235–312.
10. Patwardhan, S., Banerjee, S., Pedersen, T. (2003). Using measures of semantic relatedness for word sense disambiguation. In A. Gelbukh (ed.), *CICLing-2003: 4th International Conference on Intelligent Text Processing and Computational Linguistics*. Mexico. Lecture Notes in Computer Science. Springer.
11. Pedersen, T., Patwardhan, S., & Michelizzi, J. (2004). WordNet::Similarity—Measuring the Relatedness of Concepts. *19th National Conference on Artificial Intelligence (AAAI-2004)*, pp. 1024–1025.
12. Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. *14th International Joint Conference on Artificial Intelligence*, pp. 448–453.