Word Sense Disambiguation for Text Mining

Daniel I. MORARIU, Radu G. CREŢULESCU, Macarie BREAZU

"Lucian Blaga" University of Sibiu, Engineering Faculty, Computer and Electrical Engineering Department

Abstract:

In the field of WSD there were identified a range of linguistic phenomena such as preferential selection or domain information that are relevant in resolving the ambiguity of words. Using this information for document representation can improve the accuracy of text categorization algorithms. Mining sense of the words will bring more information in Vector Space Model representation by adding groups of words that have meaning together. In this paper present some general aspects regarding word sense disambiguation, the common used WSD methods and improvements in text categorization problem using WSD in document representation.

Keywords: *Text Mining, Word Sense Disambiguation, semantic similarity*

1 Introduction

In the recent years, significant increases in using the Web and the improvements of the quality and speed of the Internet have transformed our society into one that depends strongly on the quick access to information. The huge amount of data that is generated by this process of communication represents important information that accumulates daily and that is stored in form of text documents, databases etc. The retrieving of this data is not simple and therefore the data mining techniques were developed for extracting information and knowledge that are represented in patterns or concepts that are sometimes not obvious.

As mentioned in [3, 8], machine learning software provides the basic techniques for data mining by extracting information from raw data contained in databases. The process usually goes through the following steps:

- integration and transforming the data into a suitable format
- cleaning and selecting the data
- extracting and analyzing the rules on the data.

Machine learning techniques are divided into two sub domains: supervised learning and unsupervised learning. Under the category of unsupervised learning, one of the main tools is data clustering. This paper attempts to provide taxonomy of the most important algorithms used for clustering. For each algorithm category, we have selected the most common version of the entire family. Below we will present algorithms used in context of document clustering.

2 Unsupervised versus supervised learning

In supervised learning, the algorithm receives data (the text documents) and the class label for the corresponding classes of the documents (called labeled data). The purpose of supervised learning is to learn (discover) the concepts (rules) that correctly classify documents for given classification algorithm. Based on this learning the classifier will be able to predict the correct class for new examples. Under this paradigm, it is also possible the appearance of the over-fitting effects. This will happen when the algorithm memorizes all the labels for each case.

The outcomes of supervised learning are usually assessed on a disjoint test set of examples from the training set examples. Classification methods used are varied, ranging from traditional statistical approaches, neural networks to kernel type algorithms [5].

The quality measure for classification is given by the accuracy of classification.

In unsupervised learning the algorithm receives only data without the class label (called unlabeled data) and the algorithm task is to find an adequate representation of data distribution. Some researchers have combined unsupervised and supervised learning that has emerged the concept of semi-supervised learning [4]. In this approach is applied initially an unknown data set in order to make some assumptions about data distribution and then this hypothesis is confirmed or rejected by a supervised approach.

3 Word sense disambiguation

In the field of WSD there were identified a range of linguistic phenomena such as preferential selection or domain information that are relevant in resolving the ambiguity of words. These properties are called *linguistic knowledge sources*. Current WSD system reports does not mention these sources but rather presents low-level features such as representations of "bag-of-words" or "n-grams" used in disambiguation algorithms - one of the reasons being that the features (coding) incorporate more than one source of knowledge.

The purpose of this paper is to present the sources of knowledge and to clarify the link between sources of knowledge, features and lexical resources used in WSD.

For clarifying some relevant concepts we will present the terminology used:

Knowledge sources (KS): High level abstraction of linguistic and semantic features which are useful for resolving ambiguities, such as the domain for individual words (sports, poetry, etc..).

- Features: Different encoding types of the context. For example, the domain for one word is represented by different words that often occur with the meaning of the target word (bag-of-words) - extracted from annotated corpora or domain code included in dictionaries.
- Lexical Resources: the resources that were used to extract features.

3.1 Lexicography or "mining" the senses of a word

Lexicography is a discipline of linguistics that establishes principles and practical methods of creating dictionaries. Therefore lexicography deals with discovering senses (meanings) of words. The relationship between WSD (Word Sense Disambiguation) and lexicography is obvious, both deal with the discovery and description of the words meanings.

Because WSD has essentially the same purpose as the lexicography, we will present some lexicographical methods. The basic method used by lexicographers is called “Key word in context” (KWIC): based on some corpora the according between the different meanings of words are established. An example for the word "sense" is given in Figure 3.1

Think, and I use that word in its broadest ***sense***, I think you should jump on her. At

profoundly different light from common ***sense*** in its materialistic moment; and much

The belief that our sensations are in some ***sense*** to be understood in terms of a set of

Stage' in the development of the infant's ***sense*** of self, has no base in clinical experience

Sure ear for tonal balance and a strong ***sense*** of the orchestra's role as a virtual

Firmly rooted. </p><p> Nor is there any ***sense*** in banning strikes 'temporarily', since

of 'The Hollow Men' is furthered by a ***sense*** of confused identity. Words come to the

strengthening family life and promoting a ***sense*** of individual responsibility. Among

the top of it.) Where are the ***sense*** organs that pick up such external factors

By his fellow students. He has a great ***sense*** of humour and will keep you all welltransmission,

jams the code, prevents ***sense*** being made. The subliminal message of

free and rich, but has also begun to ***sense*** its real power. Today's West Germany

(Fig 3.1 Example of using KWIC for the word „sense")

The KWIC method can be divided into the following steps:

1. Analyze examples of KWIC for each appearance of the word.
2. Dividing the corpus lines into clusters so that the clusters items will have more common members than the other clusters.
3. For each cluster it should be expressed what features are in common.
4. Conclusions from step 3 must be coded in a dictionary specific language.

Hanks in [9] talks about two ways of extracting meaning: "Norms and exploitations". A word has a common sense and it used by the speaker most of the time. Hanks say that the sense / meanings is the norm or "potential meanings". A language is always open to interpretation, combinations and brings new "settings" of words. But a new setting can be used to create new meaning and thus the norm of the word is enriched with a new adaptation.

3.2 Knowledge Based WSD

Knowledge-based methods are a distinct category of WSD, together with the corpus-based methods. Their performance is surpassed by those based on the corpus, but they have a higher coverage (for the purposes of applicability). Usually these methods are applied for disambiguation to all words in unrestricted text while corpora-based methods are applied, in principle, to words annotated in corpora.

A simple method is quite precise if it is based on the heuristics found in the observed properties of words in texts. Such a heuristic that is used in evaluations of WSD systems is "the most common sense." The other two heuristics which we will discuss are based on a tending of word to have the same meaning in all instances of a speech (one sense per discourse) and all collocations – conversations - (one sense per collocation).

3.2.1 Most Frequent Sense

It was observed that generally one sense is predominant for words that can have several meanings, so the meaning of words can be expressed as a feature called Zipfiane distribution: a sense has a dominant frequency, the rest of the meanings have a dramatic decrease per frequency [20] Therefore, if we know the frequency of occurrence for a meaning, a simple method of disambiguation is to assign to the word the sense that exceed. This simple method is often used as a basis for comparison of WSD methods. Most systems should be more efficient than this method.

Even if this method is simple and trivial to implement, it has still a major deficiency: information about the distribution of meanings is not available in all languages, so it cannot be used for any language. In addition, if you change the text domain the frequency of the analyzed meanings it is changed also.

3.2.2 One sense per speech

This heuristic was introduced in [7]: a word tends to keep their meaning throughout the speech. It is a base rule because if we disambiguate a word, that word is disambiguated in the hole speech.

Initially this hypothesis was tested for nine words that had two meanings. These words were presented to people who had to assign the meaning to these words in 82 sentences. The obtained accuracy was 98%.

This hypothesis is true for words with "coarse grained" meanings, but in [13] is shown that for words with multiple meanings, only 33% of words had a single sense per speech.

3.2.3 One sense per collocation

This heuristic is similar to “One sense per speech” but has another purpose. It was introduced in [21] and said that a word tends to keep the same meaning when it is used in the same collocation. So the neighboring words give us clues about the meaning of a word. It was observed that this assumption is valid to neighboring words but it becomes weaker when the distance between words increases. Initial experiments showed an accuracy of 97%, and were used in experiments similar to one sense per speech, but the words had "coarse" meaning. In [1] the authors have experimented with the "fine grained" meaning of the words and obtained a decrease in accuracy (less than 70%).

An interesting aspect shown in [1] is that the meaning of the collocation remains the same when it is used corpora from different domains only that their number decreases.

3.3 Lesk Algorithm

The Lesk Algorithm [12] is one of the first algorithms developed for semantic disambiguation of words applicable to all words in unrestricted text. This algorithm needs a set of entries from a dictionary (an entry / sense) and information about the context in which the word appears. Almost any supervised WSD algorithm includes a method of overlapping contexts, that is, it calculates the distance between the ambiguous context and the specified context from the dictionary learned from the meanings from all annotations.

```
(1) for each sense i of W1
(2)       for each sense j of W2
(3)       compute Overlap(i,j), the number of words in common
                      between the definitions of sense i and sense j
(4) find i and j for which Overlap(i,j) is maximized
(5) assign sense i to W1 and sense j to W2
```

(Fig 3.2 Lesk algorithm based on a dictionary)

The pseudo code for the Lesk Algorithm is presented in Figure 3.2. The basic idea is disambiguation of a word by searching a possible overlapping from the different meanings. Given two words W_1, W_2 with the meanings defined in the dictionary N_{W1} and N_{W2}. For each pair of senses W_1^i and W_2^j, $i = 1,..., N_{W1}, j = 1,..., N_{W2}$ *we* determine the overlap by counting the number of words they have in common. Then select the pair with the largest overlapping meanings and associate this meaning to the ambiguous word.

Let the following example (taken from [12]): disambiguation of words *pine* and *cone* from the pair of words *pine cone*. The "Oxford Advanced Learner's Dictionary" defines four meanings for the word *pine*:

a) *seven kinds of evergreen tree with needle-shaped leaves*
b) *pine*
c) *waste away through sorrow or illness*
d) *pine for something, pine to do something*

and 3 senses for *cone*:

a) *solid body which narrows to a point*
b) *something of this shape, whether solid or hollow*
c) *fruit of certain evergreen trees (fir, pine)*

The first definition of the word *pine* and the third definition of the word *cone* have the most common words, namely the *tree, evergreen, pine*. Therefore, the Lesk algorithm selects as collocation sense the meanings a) and c) for *pine cone.*
The algorithm was evaluated on a set of manually annotated pairs of ambiguous words using "Oxford Advanced Learner's Dictionary" and obtained an accuracy of 50-70%.

3.3.1 Some variations of the Lesk algorithm

Since the appearance of this algorithm in 1986, there were many variations, such as:

- versions that try to solve the problem of exponential meaning growth when are considered groups of more than two words;
- variations in which each word in a given context is disambiguated individually by measuring the overlap between the dictionary definition of text and context in which the word is found;
- space meanings of words is augmented with definitions from similar words (synonyms)

3.3.2 Simulated Annealing

A major problem with the original algorithm is the exponential growth of the search space when trying to disambiguate a pair for with more than two words. The following sentence: "*I saw the man who is 98 years old and still can walk and tell jokes*" contains nine words that have more than two meanings (information extracted from WordNet): *see* (26), *man* (11), *year* (4), *old* (8), *can* (5), *still* (4), *walk* (10), *tell* (8), *joke* (3). In total there are 43,929,600 possible combinations, thus finding the optimal combination is impractical and almost impossible.

A possible solution is proposed in [6] and is called "simulated annealing". The authors define a function *E* reflecting the combination of meanings of words in a given text and whose minimum should be the correct choice of the meaning. For a given combination of senses, collect all the definitions from the dictionary. Each word that appears in the dictionary is scored with a value equal to the number of occurrences in the dictionary. Summing these scores provides the "redundancy" of the text. The function *E* is defined as the inverse function of redundancy, its purpose is to find a combination of senses that minimize this function. It begins with the initialization of a combination of senses (eg start with the most common meaning of the word). In following iterations the meaning of a word in the text is randomly replaced by another sense, the new sense replaces the initial meaning only if the value of the function *E* is reduced. The iterations are stopped when there can be no longer made other configurations with those meanings.

Tests made with this model on a set of 50 examples have shown an accuracy of 47% for fine senses and 72% for homographs.

3.3.3 The simplified Lesk algorithm

Another version that attempts to solve combinatorial explosion problem is a simplified version of the Lesk algorithm that checks each individual word with the meaning from the dictionary and assign the correct meaning according to the overlapping degree with words surrounding the ambiguous word (current context). Figure 3.3 presents simplified the steps of this variant.

```
(1) for each sense i of W
(2) determine Overlap(i), the number of words in common
        between the definition of sense i and current sentential context
(3) find sense i for which Overlap(i) is maximized
(4) assign sense i to W
```

(Fig 3.3 The simplified Lesk algorithm)

A comparison made between the simplified version and the original version show that the simplified version is much faster and accurate [19]. The evaluation was done on Senseval 2 using the version with disambiguation for all words, which had an accuracy of 58%, with 16% more accurate than the original version.

A similar version of the Lesk algorithm is used to solve the problem of word sense disambiguation using manually annotated corpora. This version, based on corpus, augments the word meanings using the context in which it appears to find new meanings by overlapping. Therefore, the chosen meaning is given by the highest value of the overlapping.

Figure 3.4 shows this algorithm. The weight of a word is defined by a metric taken from the "Information retrieval" ***IDF*** (Invers Document Frequency). Using IDF the values are weighted according to frequency of the term in the document collection using the formula:

$$IDF(t) = \log\left(\frac{1+N}{N_t}\right) \qquad (1)$$

where t is the term (word), N the number of documents in the collection and N_t the number of documents that contain term t.

The *weight(w)* is the inverse of the word frequency in the document (IDF) for the word *w* computed based on the examples from the dictionary definitions.

```
(1) for each sense i of W
(2)     set Weight(i) to 0
(3) for each [unique] word w in surrounding context of W
(4)     if w appears in the training examples or dictionary
        definition of sense i
(5)         add Weight(w) to Weight(i)
(6) choose sense i with heighest Weight(i)
```

(Fig. 3.4 Lesk algorithm based on a corpus)

3.4 Semantic similarity and semantic similarity metric

Words from a speech should be related, so that the speech makes sense - this is a natural property of language and the strongest constraint used in WSD. Words that share a common context are, in principle, related as meaning, and so therefore can be extracted the meanings using semantic distance.

There are two types of methods:

- methods based on local context, that using a limited number of words around the "target" and not taking into account additional contextual information outside this window;

- methods that use as information the global context and which are trying to create "threads of meaning" of the entire document, such as lexical chains.

Like the Lesk algorithm, these methods require computing power when there are used more than two words. But even in this case it can be applied some methods from the Lesk algorithm to reduce the computing complexity.

There are some types of metrics that quantify the extent in which two words are semantically related as well. Most of these measures are based on semantic networks and are inspired by the methodology proposed in [2] to calculate metrics between semantic networks.

We present some measures of similarity and which were tested on the WordNet hierarchy. Most of these measures are using as input a pair of definitions and return a value indicating the similarity between the two definitions.

- In [11] the minimum length is determined between two connected subsets including also the input words. In equation 2 *Path (C1, C2)* is the length of the path that connects the two concepts (the number of crossing arcs in the semantic network nodes from *C1* to *C2*).

$$Similarity(C_1, C_2) = -\log\left(\frac{Path(C_1, C_2)}{2D}\right) \qquad (2)$$

- Hirst and St-Onge [10] have introduced the concept of direction of links that forming the interconnection path. They also added the limit that the direction does not change too often. In equation 3 *C* and *k* are constants, *d* is the maximum number of changes of direction.

$$Similarity(C_1, C_2) = C - Path(C_1, C_2) - kd \qquad (3)$$

- Resnik in [17] defines the "information content", which is a measure of the specificity of a given concept. It is defined as the probability of appearance in a large corpus (equation 4)

$$IC(C) = -\log(P(C)) \qquad (4)$$

P(C) is the likelihood that the instance *C* to be encountered in the corpus. So the value of *P(C)* is higher for concepts that are higher in the hierarchy, the maximum being achieved on the top of the hierarchy (if the hierarchy contains only one value then *P(C)* will have the value 1). Using the concept of *information content* (IC), Resnik defines the semantic similarity measure between words in equation 5. This quantifies the IC of the lowest common denominator (LCS) of two concepts (i.e. the first node found in the semantic network crossing the two concepts trough the root)

$$Similarity(C_1, C_2) = IC(LCS(C_1, C_2)) \qquad (5)$$

Jiang and Conrath [14] have proposed an alternative to Resnik's equation 4, namely it calculates the difference of IC for the two concepts for computing the similarity (equation 6))

$$Similarity(C_1, C_2) = 2 * IC(LCS(C_1, C_2)) - (IC(C_1) + IC(C_2)) \qquad (6)$$

Mihalcea and Moldovan [15] found a formula that measures the semantic similarity of independent hierarchies, including hierarchies with different parts of speech. All measures defined above work only for concepts that are explicitly connected in the semantic network. The authors create virtual paths between these hierarchies using WordNet definitions. In equation 7 *C1* and *C2* represent the number of words in common found in both definitions, *descendants(C2)* is the number of concepts in the hierarchy of *C2* and *W_k* is the weight associated with each concept and is determined by the depth to which the definition is in the semantic hierarchy. It was observed that this metric works very well to disambiguation of nouns and verbs connected by syntactic relations such as verb-object, noun-adverb, etc.

$$Similarity(C_1, C_2) = \frac{\sum_{k=1}^{CD_{12}} W_k}{\log(descendents(C_2))} \qquad (7)$$

Agirre and Rigau [2] introduce the concept of "conceptual density" defined as the overlap between the semantic concept hierarchy C (root of the hierarchy) and words in the same context C. In equation 8 *m* is the total number of meanings of context C and *descendants(C)* represents the total number of concepts in the hierarchy. W_k is the weight associated with each concept in the hierarchy (*nhyp* is the number of hyponyms for the given node in the hierarchy, and the optimal value for α was determined empirically to be 0.2). To identify the target meaning in a given context, the conceptual density formula is applied to all meanings of the target word and as result will be selected the meaning that has the highest density. This concept (method) can be considered as a variation of the Lesk algorithm, the difference is that the Lesk algorithm calculates the overlap for every sense of the word, the concept of density takes into account entire sub-hierarchies that have the different meanings of the root word and computes like the Lesk algorithm the number of words in common between these sub-hierarchies and context in which the word appears, resulting that sense with the greater value. It reaches a precision of 66% in SEMCOR.

$$CD(C) = \frac{\sum_{k=0}^{m} w_k}{descendents(C)}, \quad where W_k = nhyp^{k^{\alpha}} \tag{8}$$

4 Evaluation of the algorithms

4.1 The dataset

The evaluation was made on the DSO corpus. From a total of 191 words labeled in the corpus, 21 words were selected that appear frequently in the WSD literature for testing these algorithms. They chose 13 nouns (age, art, body, car, child, cost, head, interest, line, point, states, thing, work) and 8 verbs (become, fall, grow, lose, set, speak, strike, tell) all the words are treated as separate classification problems. The number of examples per word ranged between 202 and 1482, averaging 801.1 examples per word (840.6 for the nouns and 737 for verbs). The ambiguity in this corpus is very large, the number of meanings per word being between 3 and 25, with an average of 10.1 senses per word (8.9 for nouns and 12.1 verbs).

Two types of information are used for disambiguation: *local information* and *domain information*. Be $[w_{-3}, w_{-2}, w_{-1}, \mathbf{w}, w_{+1}, w_{+2}, w_{+3}]$ the context of the words surrounding the target word w and p_i be the part of speech of the word w_i. We consider 15 patterns that refers to the local context: p_{-3}, p_{-2}, p_{-1}, p_{+1}, p_{+2}, p_{+3}, w_{-1}, w_{+1}, (w_{-2}, w_{-1}), (w_{-1}, w_{+1}), (w_{+1}, w_{+2}), (w_{-3}, w_{-2}, w_{-1}), (w_{-2}, w_{-1}, w_{+1}), (w_{-1}, w_{+1}, w_{+2}), and (w_{+1}, w_{+2}, w_{+3}). The last seven represent the collocations of two or three consecutive words. The context domain consists of the bag-of-words $\{c_1, ..., c_m\}$ which is a set of m not ordered words that appear in sentences.

The evaluated methods from this section codify features in different ways. The AdaBoost algorithm (AB) and Support Vector Machine (SVM) need binary inputs for the attributes so the local context of the attributes must be transformed into a binary representation, the attributes of the domain context remain as binary tests (occurring or not occurring domain words in the sentence). The result of this transformation is the increasing number of features to several thousand (from 1764 to 9900, depending on the word). For the Decision List algorithm (DL) it was applied the same representation as in AB and SVM.

For the Naïve Bayes (NB) and k-Nearest Neighbor (kNN) algorithms the binary representation for the attributes could not be applied so, therefore, the 15 attributes of the local context were not modified.

4.2 Corpus Experiment

An experiment to estimate the performance of this system was performed. For the classification of the examples, all methods that have been forced to have a unique meaning as output. In case of a tie the most frequently meaning is chosen.

	MFC	**NB**	**kNN**	**DL**	**AB**	**SVM**
Nouns	46,59 ± 1,08	62,29 ± 1,25	63,17 ± 0,84	61,79 ± 0,95	66,00 ± 1,47	***66,80*** ± ***1,18***
Verbs	46,59 ± 1,37	60,18 ± 1,64	64,37 ± 1,63	60,52 ± 1,96	66,91 ± 2,25	***67,54*** ± ***1,75***
TOTAL	46,55 ± 0,71	61,55 ± 1,04	63,59 ± 0,80	61,34 ± 0,93	66,32 ± 1,34	***67,06*** ± ***0,65***

(Table 4.1 Accuracy and standard deviation of learning methods*)*

Table 4.1 presents the results (accuracy and standard deviation) of all methods for the reference corpus. MFC means the Most-Frequent-Sense Classifier, a classifier that learns the most common sense from a training set. The average results are shown for nouns, verbs and total. The best results are shown in bold.

All methods perform better than the MFC with an improvement between 15 and 20.5 points. The best results were obtained by the SVM and AB (SVM is slightly better, but the difference is statistically insignificant). The AB and DL methods have the lowest accuracy. kNN is between these two extremes. Therefore, according to the pairs test *t* (*Student's t-test*), the order is as follows: SVM ≈ AB> kNN> NB ≈ DL> MFC, where A ≈ B means that A does not differ much from B, respectively A>B means that the accuracy of A is greater than that of B.

The poor performance of DL algorithm seems to contradict other previous results. One possible reason may be simply the use of a standardization method. Another reason may be that the DL is "forced" to decide with little data. Rather than force coverage of 100%, DL paradigm could be used to obtain accurate results with low coverage. (Martinez et al. in 2002 showed that the DL is in a state of very high precision but a very small coverage: 94.9% accuracy with a coverage of 9.66% and 92.79% accuracy on a coverage of 20.44%. This experiment was made on Senseval2).

In this corpus subset, the average disambiguation accuracy of nouns and verbs is almost identical. For the MFC they are almost identical (46.59%). There is a difference between the two groups of methods. The weaker methods (NB and DL) disambiguate with greater accuracy nouns than verbs, better methods like kNN, AB, SVM learn better behavior verbs and tend to a difference of one point comparing to nouns.

Schapire [18] say that AdaBoost algorithm produces poor results when trained on a small number of examples. To verify this, the authors calculated the accuracy of AB on a number of sets that have increased the number of examples per iteration. Table 4.2 shows the results of this test, making comparable with SVM.

	≤35	**35-60**	**60-120**	**120-200**	**>200**
AB	60,19	57,40	***70,21***	***65,23***	***73,93***
SVM	***63,59***	***60,18***	***70,15***	64,93	72,90

(Table 4.2 Accuracy percentage for SVM and AB for a given number of)

As expected, the accuracy of SVM algorithm is significantly better than AB using a small number of training examples per set (under 60 examples per sense). However, AB has a better accuracy on larger sets of examples (over 120 examples per sense).

In absolute terms, the overall result of the methods can be considered low (61-67%). These results increase if they could use more training examples or representation with "richer" features. However, it is known that DSO is a very ambiguous corpus and WordNet contains the meanings too "fine-grained". So the main conclusion remains: current WSD systems must be improved to be really practical.

5 Conclusions

In this paper the Authors present some general aspects regarding word sense disambiguation. Important knowledge sources and the link between this sources, features and lexical resources were presented. Also some important methods of the WSD like knowledge-based methods and corpus-based methods are presented together. In preliminary experiments the use of meaning in representation of documents for classification algorithms can improve the quality of classification result.

Acknowledgment

This work was partially supported by CNCSIS-UEFISCSU, project number PN II-RU code PD_670/2010.

6 Bibliography

[1] Agirre, E., Martínez, D., *Exploring automatic word sense disambiguation with decision lists and the Web*. Proceedings of the Semantic Annotation and Intelligent Annotation Workshop, organized by COLING. Luxembourg, 11-19., 2000;

[2] Agirre, E. Rigau, G., *Word Sense Disambiguation using Conceptual Density,* In Proceedings of the 16th International Conference on Computational Linguistics, 1996;

[3] Berkhin, P., A Survey of Clustering Data Mining Techniques, in Grouping Multidimensional Data Springer Press, pp. 25-71 2006;

[4] Bennett, K. P., Demiriz A., *Semi-supervised support vector machines,* In Advances in Neural Information Processing Systems, pages 368-374, Cambridge, MA, 1998. MIT Press;

[5] Burges, C. J. C., *A tutorial on support vector machines for pattern recognition,* In Data Mining and Knowledge Discovery, 2(2):121-167, 1998;

[6] Cowie, J., Guthrie, J., Guthrie, L*., Lexical disambiguation using simulated annealing*, In Proceedings of the 14th conference on Computational linguistics - Volume 1, Stroudsburg, PA, USA,1992;

[7] Gale, W, Church,K.,W., Yarowsky, D, *A Method for Disambiguating Word Senses in a Large Corpus*, in Computers and the Humanities, pag:415-439, 1992;

[8] Han, J., Kamber, M., *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2001;

[9] Hanks, P., *Linguistic Norms and Pragmatic Explanations, or Why Lexicographers need Prototype Theory and Vice Versa*, Papers in Computational Lexicography,1994;

[10] Hirst, G., St-Onge, D., *Lexical chains as representations of context for the detection and correction of malapropisms*, In Fellbaum 1998, pp. 305–332;

[11] Leacock, C., Chodorow, M. Miller,. G. A., *Using corpus statistics and WordNet relations for sense identification*, Computational Linguistics, Vol. no. 24-1, 1998, pp. 147–165;

[12] Lesk, M., *Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice cream cone,* In Proceedings of SIGDOC' 86, 1986;

[13] Krovetz, R., *More than one sense per discourse*, in Proceedings of SENSEVAL, Herstmonceux Castle, England, 1998;

[14] Jiang, J., J., Conrath, D. W., *Semantic similarity based on corpus statistics and lexical taxonomy*, Proceedings of 10th International Conference on Research In Computational Linguistics, 1997;

[15] Mihalcea, R., Moldovan, D., *Automatic Acquisition of Sense Tagged Corpora*, in Proceedings of Florida Artificial Intelligence Research Society Conference (FLAIRS 1999), Orlando, FL, May 1999;

[16] Rada, R., Mili, H., Bicknell, E., Blettner, M.. *Development and application of a metric on semantic nets*, IEEE Transaction on Systems, Man, and Cybernetics, 19(1): 17-30, February 1989;

[17] Resnik, P., *Using Information Content to Evaluate Semantic Similarity in a Taxonomy*, International Joint Conference for Artificial Intelligence (IJCAI-95), 1995;

[18] Schapire, R. E., *The boosting approach to machine learning: An overview. Nonlinear Estimation and Classification*. Lecture Notes in Statist. 171 149--171. Springer, New York. (2003);

[19] Vasilescu, F., Langlais, P., Lapalme, G., *Evaluating variants of the Lesk approach for disambiguating words*, in Proceedings of the Conference of Language Resources and Evaluations (LREC 2004);

[20] Zipf, G. K., *Human Behavior and the Principle of Least Effort*, Addison-Wesley, 1949;

[21] Yarowsky, D., *Word-sense disambiguation using statistical models of Roget's categories trained on large corpora*, in Proceedings of the 14th conference on Computational linguistics, Vol. 2. Association for Computational Linguistics, Stroudsburg, PA, USA, 1992;