Boosting Applied to Spanish Word Sense Disambiguation

Rocio Guillén

California State University San Marcos, San Marcos CA 92096, USA

Abstract. Recent trends in word sense disambiguation (WSD) show that the most effective approach is that of machine learning (ML). Ensemble learning methods such as boosting select a collection of hypotheses from the hypothesis space and combine their prediction. Boosting algorithms combine many weak hypotheses to find a highly accurate classification rule. In this paper we describe a system that applies a boosting algorithm to the WSD problem and present results from the SENSEVAL-3 exercise for the Spanish lexical-sample task. Our system SenseFinder utilizes Schapire and Singer's Boostexter [3] implementation of the AdaBoost.MH algorithm as the learning paradigm. We work on a set of 46 polysemous words and use tagged and lemmatized files from which we extract a window of 5 lemmas. This information is used to describe the examples and to train our system.

1 Introduction

Word sense disambiguation (WSD) involves the mapping of a given word in a text or discourse to a definition or meaning, i.e., sense, which is distinguishable from other meanings potentially assignable to that word. The task is twofold: determining all the different senses for every word, and finding a method to assign each word to the appropriate sense [2]. Sense disambiguation is essential for natural language processing tasks such as text processing, speech processing, human-computer interaction, message understanding, information retrieval and machine translation.

A wide range of symbolic and statistical or machine learning methods have been explored to solve the problem of WSD. Despite the research efforts devoted to tackle the problem there is no large-scale, broad coverage and highly accurate word sense disambiguation system available at present.

In this paper, we present the use of a machine learning approach, called boosting, to the problem of WSD. Boosting is a general method for improving the accuracy of learning algorithms. It is based on the observation that finding and combining many simple and moderately accurate "rules of thumb" is easier than finding a single, highly accurate prediction rule. [5] To find these rules, a weak or base learning method or algorithm is applied. The boosting algorithm calls the weak learner repeatedly, each time with a different distribution or weighting over the training examples. At each iteration, the weak learner generates a new weak prediction rule, and after many iterations the boosting algorithm combines the weak rules into a single prediction rule. The distribution at each iteration is chosen by placing the most weight on the examples most often misclassified by the preceding weak rules. The combination of weak rules is done by taking a weighted majority vote.

AdaBoost, a boosting algorithm introduced by Freund and Schapire [4], has been studied extensively and has been shown to perform well [7],[4],[8],[6],[5] on standard machine-learning tasks using standard machine-learning algorithms as weak learners. The first versions of the algorithm, AdaBoost.M1 and Adaboost.M2, only supported output data belonging to one class at most. Extensions to the algorithm, AdaBoost.MH and AdaBoost.MR, were designed to solve the problem of an example belonging to various classes. The goal of AdaBoost.MH is to predict all and only the correct labels, senses in our work. The goal of AdaBoost.MR is to find a hypothesis which ranks labels so that it hopefully places the correct labels (senses) at the top of the ranking. BoosTexter is a system which implements four versions of boosting based on these extensions that we have used to test our system.

AdaBoost.MH has been successfully applied to natural language processing problems such as word sense disambiguation [10], human-computer spoken-dialogue systems [11],[12] and part-of-speech tagging and prepositional phrase attachment disambiguation [13] as well as information retrieval tasks like text categorization [3] and document routing [14]. Additionally, AdaBoost has been proven to be theoretically well founded.

The paper is organized as follows. We first present in Section 2 research done in WSD using Boosting algorithms. In Section 3 we briefly describe the AdaBoost.MH algorithm and the BoosTexter system. In Section 4 we describe SenseFinder, the domain of application and evaluation metric. Experiments using the BoosTexter system are described, and results and evaluation of results are presented in Section 5. Lastly, Section 6 presents conclusions and future work.

2 Boosting algorithms for WSD

Supervised learning has become the most successful approach to tackle the problem of WSD. These algorithms follow a two-step process. The first step is to choose the representation of the context of the target word senses as a set of features. Then apply a ML algorithm to train on the chosen features and assign a sense to the target word in the test examples. Among the supervised learning algorithms that have been applied to WSD are Naive Bayes [10], Exemplar-based [15], Decision Lists [16], and Neural Networks [17].

Supervised learning algorithms suffer from high overhead for supervision and additional overhead for learning/testing when scaling to real size WSD problems. Due to this fact, research on reducing the need for supervision of corpus-based methods for WSD is currently under way. Escudero et al. [10] have worked on reducing the feature space for English. They have developed a variant of AdaBoost.MH, called LazyBoosting, which has been tested on a medium/large sense-tagged corpus containing about 193K examples of the 191 most frequent and ambiguous English words. Results showed that boosting compares favourably to the Naive Bayes and the Exemplar-based approach.

LazyBoosting is a simple modification of the AdaBoost.MH algorithm, which consists of reducing the feature space that is explored when learning each weak classifier. More specifically, a small subset S of features/attributes are randomly selected and the best weak rule is chosen among them. The idea is that if the subset S is not too small, it is more likely that a sufficiently good rule can be found at each iteration. Additionally, no feature/attribute has to be discarded thus avoiding the risk of eliminating relevant attributes.

The TALP system is based on the LazyBoosting algorithm [10]. The features represent local and topical contents and domain labels. Let w_i be the word to be disambiguated, $\dots w_{i-3}w_{i-2}w_{i-1}w_iw_{i+1}w_{i+2}w_{i+3}\dots$ the context of words around w_i, and $p_{i\pm j}$, $j = 1, 2, 3$ be the part-of-speech tag of $w_{i\pm j}$. Local context feature patterns are represented as follows.

$p_{i-3}, p_{i-2}, p_{i-1}, p_{i+1}, p_{i+2}, p_{i+3}, w_{i-2}, w_{i-2}, w_{i-1}, w_{i+1},$
$w_{i+2}, (w_{i+2}, w_{i-1}), (w_{i-1}, w_{i+1},$ and (w_{i+1}, w_{i+2})

The last three patterns represent collocations of two consecutive words.

We apply BoosTexter, an implementation of the AdaBoost.MH algorithm, to train and test the system. We use a subset of features to describe the examples and train the classifiers similar to the one described for the TALP system.

3 AdaBoost.MH

In this section we present AdaBoost.MH (see Figure 1) designed by Schapire and Singer's [1],[3],[5],[11], which is an extension to the AdaBoost algorithm. It has been designed to work for multiclass multi-label classification problems.

```
**Input:**  $(x_1, Y_1), \dots, (x_m, Y_m)$ where $x_i \in \mathcal{X}, \mathcal{Y}_i \subseteq \mathcal{Y}$
**Output:** Final hypothesis
                $f(x, l) = \sum_{t=1}^{T} h_t(x, l)$
**Initialize** $D_1(i, l) = 1/(mk)$
**For** $t = 1, \dots, T$ **do**
  1. Train weak learner using distribution $D_t$ .
  2. Weak learner returns weak hypothesis $h_t : \mathcal{X} \times \mathcal{Y} \to \Re$
  3. Update distribution
                $D_{t+1}(i, l) = \frac{D_t(i, l) exp(-Y_i[l] h_t(x_i, l))}{Z_t}$
     where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution)
```

Figure 1 AdaBoost.MH Algorithm

AdaBoost for single-label classification maintains a set of weights over training examples to force the weak learner to concentrate on examples which are hardest to classify; whereas for multiclass multi-label classification, the boosting algorithm maintains a set of weights and labels. Training examples and their corresponding labels that are harder to classify get incrementally higher weights. The intention is to force the weak learner to focus on examples and labels that will contribute more to the overall goal of finding a highly accurate classification rule.

Let S be a sequence of training examples $< (x_1, Y_1), \ldots, (x_m, Y_m) >$ where each x_i belongs to some domain or instance space $\mathcal{X}$ and each label Y_i is in some label set $\mathcal{Y}$. Assume that $\mathcal{Y}= \{ -1, +1 \}$. As described above, AdaBoost.MH maintains a set of weights as a distribution D_t over examples and labels. Initially, this distribution is uniform. On each iteration t, the distribution is inputted to the weak learner to compute a weak hypothesis h_t. The output of the weak learner is a hypothesis $h : \mathcal{X} \times \mathcal{Y} \rightarrow \Re$. Let l be a label, then the sign of $h(x, l)$ is interpreted as a prediction of the value of $Y[l]$, i.e., whether l is or is not assigned to x. The magnitude of the prediction denoted $\mid h(x, l) \mid$ is interpreted as a measure of confidence in the prediction. The distribution D_t is updated in a manner that increases the weight of example-label pairs (x, l) which are misclassified by h_t. Testing the value of a Boolean predicate and making a prediction based on that value is carried out using very weak hypotheses for WSD following [13] and [10]. The predicates in our system are of the form $f\ v$ where f is a feature (word) and v is its corresponding part-of-speech and lexical features, e.g., *arte NCCS000*. The predicates used are $f_{i-2}\ v_{i-2}$, $f_{i-1}\ v_{i-1}$, $f_i\ v_i$, $f_{i+1}\ v_{i+1}$, $v_{i+2}\ v_{i+2}$. Formally, based on a given predicate P, we are interested in weak hypotheses h which make predictions as follows.

$$h(x, l) = \begin{cases} c_{0t} \text{ if } P \text{ holds in } x, & c_{jt} \in \Re \\ c_{1t} \text{ otherwise}, & c_{jt} \in \Re \end{cases}$$

3.1 BoosTexter

In practice, the AdaBoost.MH algorithm has been implemented as the BoosTexter system for text categorization tasks. BoosTexter works with data which may be of various forms. In general, each instance or example is split up into multiple fields. These fields may be one of the following four types: continuous-value attribute (e.g., *word-position*), discrete-valued attribute (e.g., *part-of-speech*, text string (e.g., *actuar en Cannes*), or scored-text string (e.g., *word-frequency*).

BoosTexter combines many simple hypotheses (rules) iteratively. Each hypothesis (rule) consists of a simple binary test and predictions for each outcome of the test. Depending on the type of input field the binary test has one of the following forms.

Type	**Test**
Discrete	Does the attribute have a particular value?
Continuous	Is the value of the attribute > threshold or attribute < threshold?
Text	Is the string (ngram) in given text?
Scored	Is the score of the word > threshold or is the score < threshold?

The predictions associated with each outcome of a given hypothesis (rule) are described by a set of weights over the possible labels. The weights should not be interpreted as probabilities.

4 SenseFinder System

The SenseFinder system was developed for the SENSEVAL-3 Spanish Lexical Sample Task [18]. The purpose of this task is to evaluate supervised and semi-supervised learning algorithms for WSD. Experiments are carried out on a set of 46 words. The examples for all 46 words in both the training and test set have been extracted from the year-2000 corpus of the Spanish EFE News Agency. Each example has been tagged with a unique sense. Additionally, POS tagged and lemmatized files have been provided, in which the contexts of the examples are tokenized, lemmatized and POS tagged.

The system utilizes BoosTexter for describing the examples, training and testing since the WSD problem can be considered as a categorization task in which a word is assigned to a pre-existing set of senses. Our goal is to generate a classifier for each word; each word represents a categorization problem.

The first step before applying the boosting algorithm is to extract the information to describe the examples used for training. Extraction is done by tokenizing the tagged and lemmatized file looking for the word tagged as the head. The next step consists of creating a window that includes the following: withe previous two lemmas and their corresponding part-of-speech; the head's lemma and corresponding part-of-speech; and the following two lemmas and their corresponding part-of-speech. Punctuation marks are ignored and diacritic (accent) marks are removed. A sample input sentence, part of its corresponding tagged sentence, and generated output is shown in Figure 2. Output thus generated become the training examples. The same step is applied to the test set to generate the examples for testing.

Input Sentence:
Photopainters.com, según los jóvenes empresarios, "no es estrictamente comercial o de arte, sino una web que une cultura popular, <head>arte</head> y tradición".

Input Tagged sentence:
<w frm="cultura" lem="cultura" pos="NCFS000"/>
<w frm="popular" lem="popular" pos="AQ0CS0"/>
<w frm="," lem="," pos="Fc"/>
<w frm="arte" lem="arte" pos="NCCS000" head="yes"/>
<w frm="y" lem="y" pos="CC"/>
<w frm="tradición" lem="tradición" pos="NCFS000"/>

Output:
cultura NCFS000, popular AQ0CS0, arte NCCS000, y CC, tradicion NCFS000

Figure 2 Sample Sentence

Next, a file with training examples for each word is fed to the BoosTexter system. Once training is completed, a file with the examples from the test set for each word is fed to the BoosTexter system. Lastly, the individual test set files are concatenated to be evaluated.

4.1 Scoring Scheme

Evaluation of the results has been carried out by SENSEVAL-3 using an official scorer proposed by Resnik and Yarowsky [19] and two other evaluations for the particular task. The official scorer is derived by assigning probabilities over sense labels generated by WSD algorithms. Given a probability distribution over sense labels and a single known-correct sense label, the algorithm's score should be the probability that the algorithm assigns to the correct sense label. One of the other scoring schemes does a more complete evaluation, including word-by-word results and results by groups of words. Words are grouped by part-of-speech, i.e., noun (n), verb (v), adjective (a), and by the accuracy of the Most-Frequent-Sense baseline classifier. The third scoring scheme takes the results of a baseline system that always assigns to each word the most frequent sense according to the training set.

5 Experiments

In our experiments we work with 46 datasets, one per word to be trained. The number of classes, i.e., senses ranges from 2 to 8, the number of training examples ranges from 69 to 268; the number of examples in the test set is over half the number of training examples (see Table 1).

Word	POS	Sen	Train	Test	Word	POS	Sen	Train	Test	Word	POS	Sen	Train	Test
		ses	Ex.	Ex.			ses	Ex.	Ex.			ses	Ex.	Ex.
actuar	v	3	133	67	corona	n	3	124	64	partido	n	2	133	66
apoyar	v	3	259	128	duplicar	v	2	254	126	pasaje	n	4	220	111
apuntar	v	4	213	106	explotar	v	5	212	103	perder	v	4	218	106
arte	n	3	251	121	ganar	v	3	237	118	popular	a	3	133	67
autoridad	n	2	268	132	gracia	n	3	72	38	programa	n	3	267	133
bajar	v	3	235	115	grano	n	3	117	61	saltar	v	8	200	101
banda	n	4	230	114	hermano	n	2	128	66	subir	v	3	231	114
brillante	a	2	126	63	jugar	v	3	236	117	simple	a	3	117	61
canal	n	4	262	131	letra	n	5	226	114	tabla	n	3	130	64
canalizar	v	2	253	126	masa	n	3	172	85	tocar	v	6	158	78
ciego	a	3	102	52	mina	n	2	134	66	tratar	v	3	143	72
circuito	n	4	261	132	natural	a	5	215	107	usar	v	2	263	130
columna	n	7	129	64	naturaleza	n	3	258	128	vencer	v	3	134	65
conducir	v	4	134	66	operación	n	3	134	66	verde	a	2	69	33
corazón	n	3	123	62	órgano	n	2	263	131	vital	a	2	131	65
										volar	v	3	122	60

Table 1 Set of 46 words

The total number of training examples used was 8430, and the total number of test examples was 4195.

The binary-valued attributes for describing the examples correspond to five features which constitute a very narrow linguistic context. The five features which are a modification of those used in [10] are l_{i-2} pos_{i-2}, l_{i-1} pos_{i-1}, l_i pos_i, l_{i+1} pos_{i+1}, l_{i+2} pos_{i+2}, l is a lemma and pos is its corresponding part-of-speech.

We trained the system using BoosTexter for each word a different number of iterations depending on the number of examples for a word in the training set and the different senses attributable to the word. The assumptions made are: the more examples to be trained, the more attributes need to be examined to determine a weak rule and the more senses for a word the higher the probability of a word to be misclassified. The total number of iterations and the number of hypotheses are generated per word during the learning process. Once the training was completed we ran Boostexter on the test set. The results generated were submitted to SENSEVAL-3 for evaluation.

Overall results of the official evaluation, the POS-based evaluation, and the Most Frequent Sense (MFS) evaluation for the test set are shown respectively in Table 2, Table 3, and Table 4.

precision	74.09%	3108 correct of 4195 predictions
recall	74.09%	3108 correct of 4195 in total
F1 score	74.09	F1 = (2*precision*recall)/(precision+recall)
coverage	100.00%	4195 examples predicted of 4195 in total

Table 2 Overall Results

POS	total	predicted	hit	coverage	precision	recall	F1
a	448	448	347	100.00	77.46%	77.46%	77.46
n	1949	1949	1440	100.00	73.88%	73.88%	73.88
v	1798	1798	1321	100.00	73.47%	73.47%	73.47

Table 3 POS-based Average Evaluation

word-group	total	predicted	hit	coverage	precision	recall	F1
1.MFS>95	635	635	608	100.00	95.75%	95.75%	95.75
2.MFS90-95	429	429	394	100.00	91.84%	91.84%	91.84
3.MFS80-90	374	374	325	100.00	86.90%	86.90%	86.90
4.MFS70-80	618	618	394	100.00	63.75%	63.75%	63.75
5.MFS60-70	523	523	361	100.00	69.02%	69.02%	69.02
6.MFS50-60	586	586	390	100.00	66.55%	66.55%	66.55
7.MFS40-50	673	673	456	100.00	67.76%	67.76%	67.76
8.MFS<40	357	357	180	100.00	50.42%	50.42%	50.42

Table 4 Most Frequent Sense

Analysis of individual results for each word show that the number of features chosen for training needs to be increased to include more features. Additional syntactic features and/or prior knowledge are necessary to improve recall and precision of some words such as the nouns *letra, columna, gracia* and verbs such as *perder, conducir, tocar, saltar*. In the case of the nouns *letra* and *columna*, the number of senses is 5 and 7 respectively, which makes the disambiguation task more difficult; but it also may be the case that the training examples given do not contribute to generate a highly accurate hypothesis. A similar situation occurs with the verbs *saltar* with 8 senses, *tocar* with 6 senses and *perder* and *conducir* with 4 senses each. Results including the top five and the bottom five are presented in Table 5.

word	total	predicted	hit	coverage	precision	recall	F1
usar.v	130	130	127	100.00	97.69%	97.69%	97.69
canalizar.v	126	126	122	100.00	96.83%	96.83%	96.83
autoridad.n	132	132	127	100.00	96.21%	96.21%	96.21
duplicar.v	126	126	121	100.00	96.03%	96.03%	96.03
hermano.n	66	66	62	100.00	93.94%	93.94%	93.94
conducir.v	66	66	36	100.00	54.55%	54.55%	54.55
gracia.n	38	38	19	100.00	50.00%	50.00%	50.00
columna.n	64	64	31	100.00	48.44%	48.44%	48.44
perder.v	106	106	50	100.00	47.17%	47.17%	47.17
letra.n	114	114	50	100.00	43.86%	43.86%	43.86

Table 5 Results for 10 words

6 Related Work

Some of the supervised learning approaches applied in the Spanish Lexical Sample task were an exemplar based classifier [20], support vector machines [21], decision trees [24], pattern abstraction, and kernel methods [22] and a combination of three classifiers [23]. The use of these approaches in the Spanish Lexical Sample Task is briefly described next.

The exemplar-based classifier measures the similarity between a new instance and the representation of some labelled examples. The terms are represented as bags of contexts. Words, lemmas and senses are represented in the same space, called Context Space, where similarity measures can be defined. In the SVM approach each training and test item is represented as a feature with weights; its dimensions correspond to properties of the context. A family of SVM classifiers was constructed for the senses of each word. All positive training examples for a word sense were treated as negative examples for all the other senses. The decision trees approach uses an ensemble of three bagged decision trees. It is based on the idea that different views of the training examples for a given target word will result in classifiers that make complementary errors. Thus their combined performance will be better than individual performances. Pattern abstraction is a methodology which uses different knowledge sources to extract information. This represents a limitation that has been solved with kernel methods. Kernels are similarity functions between instances that allow the integration of different knowledge sources and the modelling of linguistic features in SVM. The combination of classifiers included a nearest-neighbor clustering classifier, a naive Bayes classifier, and a decision list classifier; each one was trained on several permutations of the extracted feature set, then the answers were combined using voting.

7 Conclusion and Future Work

Our precision, recall and F_1 measure were very close to the Most Frequent sense Classifier (MFC) scores reported by the task organizer. Compared with the supervised learning techniques presented, the boosting algorithm applied to WSD of Spanish did not perform well. We found some inconsistencies in the results for some words and are repeating experiments for those words. We have no conclusive results to report in this paper due to time constraints. Compared with unsupervised learning techniques, the boosting algorithm performed better. A detailed description of the results and system comparisons appears in [25]. Further research includes the following tasks: testing the algorithms in other Spanish tagged corpora; implementing, comparing and evaluating other supervised learning approaches; and adding syntactic features.

References

1. Schapire, R., Singer, Y.: BoosTexter:A Boosting-based System for Text Categorization *Machine Learning*, **38**(2/3):135-168, 2000.
2. Ide, N., Veéronis, J.: Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Computational Linguistics*, **24**(1):1-41, 1998.
3. Schapire, R.E.: The Boosting Approach to Machine Learning An Overview. *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.
4. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1):119-139, August,1997.
5. Schapire, R.E., Singer, Y.: Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, **37**(3):297-336,1999.
6. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, 148-156, 1996.
7. Bauer, E., Kohavi, R.: An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants. *Machine Learning Journal. Special Issue on IMLM for Improving and Scaling Machine Learning Algorithms*, **36**(1-2):105-139,July/August,1999.
8. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139-158.
9. Quinlan, J. R.: Bagging, boosting and C4.5. *Proceedings of the 13th National Conference on Artificial Intelligence, AAAI*, 1996.

10. Escudero, G., Màrquez, L., Rigau, G.: Boosting Applied to Word Sense Disambiguation. *Proceedings of the 12th European Conference on Machine Learning, ECML 2000*, Barcelona, Spain.
11. Schapire, R.E.: Advances in Boosting. *Uncertainty in Artificial Intelligence: Proceedings of the Eighteenth Conference*, 2002.
12. Rochery, M., Schapire, R., Rahim, M., Gupta, N., Riccardi, G., Bangalore, S., Alshawi, H., Douglas, S. : Combining Prior Knowledge and Boosting for Call Classification in Spoken Language Dialogue. *Intrnational Conference on Acoustics, Speech and Signal Processing*, 2002.
13. Abney, S., Schapire, R.E., Singer, Y.: Boosting Applied to Tagging and PP Attachment. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.
14. Iyer, R., Lewis, D., Schapire, R.E., Singer, Y., Singhal, A. Boosting for Document Routing. *Ninth International Conference on Information and Knowledge Management*, 2000.
15. Fujii, A., Inui, K., Tokunaga, T., Tanaka, H.: Selective Sampling for Example-based Word Sense Disambiguation. *Computational Linguistics*, **24**(4):573-598, 1998.
16. Yarowsky, D.: Decision Lists for Lexical Ambiguity Resolution: Application ro accent restoration in Spanish and French. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, ACL*, 1994.
17. Towell, G., Vorhees, E.M.: Disambiguating Highly Ambiguous Words. *Computational Linguistics*, **24**(1):125-145, 1998.
18. SENSEVAL-3: URL *http://www.senseval.org/senseval3*
19. Resnik, P., Yarowsky, D.: A perspective on word sense disambiguation methods and their evaluation, position paper presented at the *ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, 1997
20. Artiles, J., Peãs, A., Verdejo, F.: Word Sense Disambiguation based on Term to Term Similarity in a Context Space. *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004
21. Cabezas, C., Battacharya, I., Reskink, P.: The University of Maryland SENSEVAL-3 System Descriptions. *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
22. Strapparava, C., Gliozzo, A., Giuliano, C.: Pattern Abstraction and Term Similarity for Word Sense Disambiguation: IRST at Senseval-3. *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
23. Wicentowski, R., Thomforde, E., Packel, A.: The Swarthmore College SENSEVAL3 System. *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
24. Pedersen, T.: The Duluth Lexical Sample Systems in SENSEVAL-3: *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
25. Màrquez, L., Taulé, M., Martí, M.A., García, M., Artigas, N., Real, F.J., Ferrés, D. SENSEVAL-3: The Spanish Lexical Sample Task. *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.