Learning Rules for Large-Vocabulary Word Sense Disambiguation: a comparison of various classifiers

Georgios Paliouras, Vangelis Karkaletsis,
Ion Androutsopoulos, and Constantine D. Spyropoulos

Institute of Informatics & Telecommunications, NCSR "Demokritos",
Aghia Paraskevi Attikis, Athens, 15310, Greece
`{paliourg,vangelis,ionandr,costass}@iit.demokritos.gr`

Abstract. In this article we compare the performance of various machine learning algorithms on the task of constructing word-sense disambiguation rules from data. The distinguishing characteristic of our work from most of the related work in the field is that we aim at the disambiguation of all content words in the text, rather than focussing on a small number of words. In an earlier study we have shown that a decision tree induction algorithm performs well on this task. This study compares decision tree induction with other popular learning methods and discusses their advantages and disadvantages. Our results confirm the good performance of decision tree induction, which outperforms the other algorithms, due to its ability to order the features used for disambiguation, according to their contribution in assigning the correct sense.

1. Introduction

The meaning of a word may vary significantly according to the context in which it is used. For instance the word "bank" will have a completely different meaning in financial text than in geological text. This is a case of a clearly identifiable sense distinction, but there are cases where different senses of a word may be harder to distinguish, e.g. "bank" as a financial institution and as a building. Both senses are likely to appear in the same context and one needs to take into account the details of their use, in order to distinguish between them. The process of distinguishing between different senses of a word is called word-sense disambiguation (WSD). Word-sense disambiguation is necessary for a number of tasks in natural language processing (NLP), such as machine translation, query-based information retrieval and information extraction.

In general, the rules for distinguishing between the senses of different words differ. For instance, a valid disambiguation rule for the senses of the word "bank" would examine the occurrence of the words "river", "financial", etc. in the context of the

ambiguous word. This evidence would be completely irrelevant for most other words. Thus the disambiguation rules are in general word-specific. Furthermore, it is difficult to construct such rules manually, especially when the difference between the senses is not great, e.g. "bank" the institution and the building. For this reason, the automatic construction of disambiguation rules is highly desirable. One way to achieve this aim is by applying machine learning techniques to training data containing the various senses of the ambiguous words.

An important aspect of the work presented here, as compared to related work, is that all content words (rather than a handful of them) in the training texts are subject to disambiguation. This step towards large-vocabulary disambiguation is necessary if WSD systems are to be used in practice. However, the automatic construction of large-vocabulary disambiguators is hard, due to the sparseness of the training data for each individual word. An important issue that arises in this context is the ability of the learning method to construct simple general rules that apply to all words, capturing regularities in less frequent words in the data.

In an earlier study [15], we have shown that good results on the task of large-vocabulary disambiguation can be achieved with the use of decision tree induction. In the study presented here we compare decision tree induction with a variety of other learning methods. All of these methods are general-purpose, i.e., they were not designed for this particular task and they are implemented in the platform WEKA, which is publicly available for research purposes.[1]

Section 2 presents related work in WSD. The WSD task, as this is realised in our approach, is presented in Section 3. The learning methods that are used in the study are briefly presented in section 4. Section 5 presents our experiments (i.e., experimental set-up and results). Finally, in section 6, we summarise the work and present our future plans.

2. Related Work

Early efforts in automating the sense disambiguation task made use of Machine-Readable Dictionaries (MRDs) and thesauri, which associate different senses of a word with short definitions, examples, synonyms, hypernyms, hyponyms, etc. A simple approach of this type is to compare the dictionary definitions of words appearing in the surrounding text of an ambiguous word with the text in the definition of each sense of the ambiguous word in the dictionary. Clearly, the higher the overlap between the dictionary definitions of the surrounding words and the definition of a particular sense of the ambiguous word, the more likely it is that this is the correct sense for the word. Some of the methods that are based on MRDs and thesauri are presented in [11], [19], [4]. The resources that are commonly used in these studies are: the WordNet, Longman's Dictionary of Contemporary English (LDOCE),

---

[1] URL: http://www.cs.waikato.ac.nz/ml/weka/index.html

Roget's thesaurus and Collins English Dictionary (CDE). A more thorough account of this work can be found in [7].

Despite the useful information that they contain, MRDs and thesauri are often inadequate for WSD, e.g. MRD sense definitions are often non-representative of the context in which the sense is met. As a result, the focus of WSD research has recently turned to *corpus-based* methods. According to this approach, a corpus of text is used as training data for the construction of disambiguation rules for different words. The construction of these disambiguation rules is achieved by a variety of machine learning methods.

An important distinguishing feature for machine learning methods is the extent of supervision provided for training. Supervision is provided in the form of hand-labelling the examples that are used for learning. In the case of WSD, a fully supervised method requires that all occurrences of an ambiguous word in the training text be labelled with the correct sense. The sense labels are typically taken from a dictionary. Given this information, a supervised learning algorithm constructs rules that achieve high discrimination between occurrences of different word-senses. Examples of supervised learning methods for WSD appear in [1], [6], [9], [22], [18]. The learning methods used in those studies are general-purpose, including: decision-tree induction, decision-list induction, feed-forward neural networks with backpropagation and naïve Bayesian learning. Their results are very encouraging, exceeding 90% correct sense labelling in some cases.

However, this high disambiguation rate is achieved at the expense of disambiguating only a small number of words. In all of the above-mentioned studies only a handful of words are included in the evaluation experiments and for each of these words a sufficient number of examples are provided, covering all senses of the word. This is an unrealistic scenario, when aiming to construct a system to be used in practice. The results presented here are on a much larger scale, considering all content words of a corpus.

In addition to the supervised approaches to learning WSD systems, unsupervised learning has been used for the same purpose, which does not require hand-tagging of the training data, e.g. [21], [10], [17]. As expected, the performance of the unsupervised learning approaches is lower than that of their supervised counterparts. However, performance evaluation of unsupervised learning methods is not straightforward, as there are no correct tags against which to compare the results of the disambiguation.

A compromise solution between supervised and unsupervised learning is the use of a small number of tagged examples, together with a large set of untagged data. Such partially supervised learning methods are presented in [23], [18], using rule-learning and neural networks respectively.

An important issue for any WSD learning algorithm is what features will be used to construct the disambiguation rules, i.e., what evidence is relevant for WSD. Since syntactic information is not considered useful for hard WSD tasks, the evidence commonly used consists of words that can be found in the neighbourhood of the ambiguous word. The question that arises then is how large this neighbourhood ought to be, i.e., how broad a context is needed for disambiguation. According to this

criterion, the WSD methods in the literature can be divided into two large groups: *local* and *topical* WSD. In local WSD only the close neighbourhood of the word (<10 words on each side) is used. Topical methods on the other hand use a larger context window (> 50 words on each side). None of the fairly recent approaches presented above uses purely local information. Yarowsky [21] and Schütze [17] present purely topical methods, but in both papers the value of local information is noted. Most of the recent approaches, e.g. [22], [18], combine local and topical information, in order to improve their performance.

A critical component of any application of machine learning is the representation of the training examples and the generated model, i.e., the disambiguator here. The most popular representation for training examples in machine learning is the feature vector, i.e., a fixed set of features, taking values from a finite set. Examples of such features in WSD may be collocated words, within a window surrounding the ambiguous word. These types of feature dominate the literature on learning methods for WSD. Despite the encouraging results obtained in most studies, features of this type cause a combinatorial explosion in the space of possible solutions. This is due to the unbounded value set of the features, e.g. almost any word can be a collocate of any other word. Little work has been done so far on alternative forms of representation. Yarowsky [21] looks at classes of words and Schütze [17], groups words that occur in similar contexts.

3. The Word Sense Dismabiguation Task

The data used in this study are extracted from the SEMCOR corpus, which is a selection of various texts from the Brown corpus. The important feature of this corpus is that the content words, i.e., nouns, verbs, adjectives and adverbs, have been hand-tagged with syntactic and semantic information, as part of the WordNet project. A subset of SEMCOR is used here containing only financial news articles. We have chosen this subset of SEMCOR, because a previous study [14] has shown that better disambiguation performance can be achieved by focusing on a specific thematic domain.

The SEMCOR corpus is tagged with WordNet sense-numbers. However, the information extraction system for which we want to use the WSD methods,[2] makes use of the Longman Dictionary of Contemporary English (LDOCE). For this reason we translated the WordNet tags into their equivalent in LDOCE. This translation was supported by a resource that was constructed in the WordNet project: a mapping between the senses in the two dictionaries [2]. The mapping between WordNet and LDOCE senses suffers in several respects. Two important problems are:

- there is a large number of senses in both dictionaries that have not been mapped onto senses in the other dictionary;
- the mapping between senses is hardly ever one-to-one, e.g. seven different Wordnet senses for the verb 'absorb' are mapped onto the same LDOCE sense, while the word has four LDOCE senses.

Due to these problems, there is a loss of information in the translation of the data from WordNet to LDOCE tags. In average, only a quarter of the words in the corpus can be assigned LDOCE senses.

The SEMCOR text is translated into training data using the feature-vector representation. For each word, each of its LDOCE senses with the correct part of speech is represented as a separate example case for learning. The correct sense is labeled as a positive example and all other senses as negative. Each example case contains the following characteristic information about the word and the context in which it appears: the lemma of the word, the rank of the sense in LDOCE corresponding to how frequently the sense appears in general text, the part-of-speech tag for the word and ten collocates (first noun/verb/preposition to the left/right and first/second word to the left). For instance, the word "bank" in the following example:

If you destroy confidence in banks you do something to the economy.

is being used with LDOCE sense rank 1. The feature vector representation of the positive example extracted from this sentence is:

{"bank", 1, noun, "confidence", "destroy", "in", "economy", "do", "to", "in", "confidence", "you", "do"}

where "bank" is the lemma, 1 is the sense rank, noun the part of speech, "confidence", "destroy", "in", the noun, verb, preposition on the left, "economy", "do", "to", similarly on the right and "in", "confidence", "you", "do", the four words surrounding the word "bank". In addition to this positive example a number of negative vectors are generated, one for each alternative sense for the noun "bank".

The evidence used in the disambiguation of different words differs significantly. This is because the disambiguation evidence consists of specific words that commonly occur in the context of the ambiguous word. Due to this fact, almost all work in WSD has considered words individually, i.e., a different disambiguator is built for each word. In the work presented here we test this hypothesis by attempting to construct a common disambiguation system for all content words in SEMCOR, using the lemma of the ambiguous word as a feature in the training examples. Depending on the use of the lemma by the disambiguation system, we can judge whether the system performs word-specific disambiguation, or whether the learning procedure has identified word-independent patterns in the data. The identification of such general patterns is particularly useful when disambiguating rare words, for which there is insufficient training information in the data. This is unavoidable when performing large-vocabulary WSD.

4. Learning methods

The main aim of this article was to investigate the applicability of different machine learning methods on the task of large-vocabulary disambiguation. In order to achieve that, we wanted to examine as many different algorithms as possible, under the same conditions. For this purpose we chose the experimental system WEKA, which provides a large number of algorithms on a common platform. Furthermore, it incorporates useful experimental tools, such as k-fold cross-validation. WEKA is publicly available for research purposes and all the algorithms in the system are presented in detail in [20].

The algorithms that were chosen for this study can be grouped into three broader categories: *symbolic induction algorithms*, *probabilistic classifiers* and *memory-based classifiers*. Symbolic induction algorithms, use heuristic search methods to construct symbolic classification models, which generalise the information in the training data. In a similar vein, probabilistic classifiers build probabilistic classification models, based on a statistical analysis of the training data. Finally, memory-based classifiers do not perform considerable generalisation of the data. They keep the training data in a well-organised memory and classify unseen instances, by finding the most-similar training instances in memory.

Each of the three categories is represented by its most popular members. Thus, we have used: two symbolic induction algorithms (decision tree and decision rule induction), one probabilistic classifier (naïve Bayesian) and two memory-based classifiers (decision table and instance-based). All five of these algorithms have been applied successfully in a variety of other tasks and it has been observed that none of them is universally better or worse than others. The performance of the algorithms varies significantly, according to the nature and the representation of the problem. Thus, our aim is not to evaluate the quality of the algorithms as such, but to examine which of them are suitable to the WSD task, represented as explained in section 3.

Decision tree induction (J48). The decision tree induction algorithm used here is called J48 and it is an improved version of the C4.5 algorithm [16] that we used in [15]. Given example cases in the format described in section 3, J48 constructs a decision tree, which can then be used to assign sense tags to unseen data. J48 generates decision trees, the nodes of which evaluate the descriptive features of words, i.e., the lemma, sense-rank, part-of-speech tag and the values of collocates. Following a path from the root to the leaves of the tree a sequence of such tests is performed, resulting in a decision about the appropriate sense for the word. The decision trees are constructed in a top-down fashion, by choosing the most appropriate feature each time. The features are evaluated according to an information-theoretic measure, which provides an indication of the "classification power" of each feature. Once a feature is chosen, the training data are divided into subsets, corresponding to different values of the selected feature, and the process is repeated for each subset, until a large proportion of the instances in each subset belong to a single class. This process is also known as "iterative dichotomization", although it usually produces multi-branch (i.e., not binary) splits at each node. J48 provides an

option for binary splits, which we evaluate separately here. The binary-split version of J48 will be henceforth called J48b.

Decision rule induction (PART). Large decision trees can become incomprehensible. For this reason, a number of algorithms have been devised that construct a list of decision rules for classification, rather than a tree. Some of these algorithms, such as AQ15 [3] and CN2 [12], construct decision rules directly from the training data. However, it is also quite straightforward to construct rule lists from decision trees. Each path from root to leaves is a conjunctive rule, the conditions of which are the individual nodes. Alternative paths are combined disjunctively. This naïve approach can be improved by optimizing the structure of the resulting rules. This process was first introduced in the program C4.5rules [16], which translates decision trees that have been constructed by C4.5 into rule lists. The rule lists that were constructed by C4.5 were even shown to achieve better classification performance than the original trees in some cases. The algorithm that we use here is a variant of this approach and is called PART. It translates trees that are generated by J48 into rule lists.

Naïve Bayesian classification (NB). The Naïve Bayesian [5] is arguably one of the simplest probabilistic classifiers. The model constructed by this algorithm is a set of probabilities. Each member of this set corresponds to the probability that a specific feature value f_i appears in the instances of class c, i.e., $P(f_i|c)$. These probabilities are estimated by counting the frequency of each feature value in the instances of a class in the training set. Given a new instance, which assigns specific values to the descriptive features, e.g. lemma, sense-rank, part-of-speech tag and collocates, the classifier estimates the probability that the instance belongs to a specific class, based on the product of the individual conditional probabilities for the feature values in the instance. The exact calculation uses Bayes theorem and this is the reason why the algorithm is called a Bayesian classifier. The algorithm is also characterized as Naïve, because it makes the assumption that the features are independent given the class, which is rarely the case in real-world problems. Despite this strong assumption, the algorithm performs surprisingly well on a range of tasks.

Instance-based classification (IBk). Instance-based (or memory-based) classifiers [13] do not build general models from the training data. Instead of that, they store *all* training instances in a memory structure, and use them directly for classification. The simplest form of memory structure is the multi-dimensional space defined by the features in the instance vectors. Each training instance is represented as a point in that space. The classification procedure is usually a variant of the simple *k*-nearest-neighbor (*k*-nn) algorithm. *k*-nn assigns to each new unseen instance the majority class among the *k* training instances that are closest to the unseen instance (its *k-neighborhood*). Euclidean distances are typically used to measure the similarity between two instances. The implementation of the *k*-nn algorithm in WEKA is called IBk and provides several parameters, which can be used to improve the performance of the algorithm, e.g. weighting the instances in the neighbourhood according to their distance from the test instance, when deciding on the class of the test instance.

Decision table classification (Dtable). Decision tables [8] are reduced versions of the original training set. The data set can be viewed as a table, the rows of which

correspond to different instances and the columns to the different features. Reduction is done in both directions, by removing instances and features. The end-result of this process is a smaller table, which can be used for classification in a similar manner as the original training set in the *k*-nn algorithm. The only difference is that some of the features in the test instances are ignored, as they have been removed from the table. The selection of instances and features to be kept in the decision table is usually based on information theoretic measures, as in the inductive learning algorithms. In this manner, a decision table can also be viewed as a list of decision rules, all of which have the same length.

5. Experimental Results

5.1 Experimental set-up

The measures that were chosen for the evaluation of the algorithms are those typically used in the language engineering literature: *recall* and *precision*. The recall measure is based on the number of positive examples that were identified as such by the WSD system. These are called True Positives (TP). The actual recall measure is the ratio of True Positives to the total number of positives (P) in the test data:

$$recall = \texttt{TP/P}.$$

On the other hand, precision is the ratio of True Positives to all examples classified as positive, i.e., True Positives (TP) plus False Positives (FP) by the system:

$$precision = \text{TP/(TP+FP)}.$$

In all of the figures presented in the following sections, the performance of the system is measured on unseen data. Furthermore, in order to arrive at a robust estimate of performance, we use ten-fold cross-validation at each individual experiment. According to this experimental method, the data set is divided into ten equally sized pieces and the learning algorithm is run ten times. Each time, nine of the ten pieces are used for training and the tenth is kept as unseen data for the evaluation of the algorithm. Each of the ten pieces acts as the evaluation set in one of the ten runs. Thus, each recall and precision figure presented in the following sections is an average over ten runs, rather than a single train-and-test result, which can often be accidentally high or low. Finally, it should be noted that, unlike MRD-based approaches, the constructed disambiguators make no use of external resources.

5.2 Results

The financial news articles of SEMCOR consist of 3,613 word occurrences, of which 1,987 have been tagged with WordNet senses, resulting in 753 word occurrences with LDOCE senses and 355 distinct words. The LDOCE polysemy of the dataset is 3,516/753=4.67, and the ratio of word occurrences to distinct words,

i.e., the average word repetition is 753/355=2.12. Word repetition is one indication of the richness of the vocabulary in the text. The closer the ratio is to 1, the richer the vocabulary.

Table 1 presents the results for the algorithms examined in this study. In order to set the results in context, we examine the following simple base case: we consider as appropriate the first sense of each word in LDOCE, i.e., the most frequently used sense. The shaded row in Table 1 shows the performance of this simple rule. Clearly, any results close or below the performance of this simple rule are not acceptable as a solution to the problem.

Table 1. Performance of different classifiers on large-vocabulary word-sense disambiguation.

Classifier	Recall	Precision
J48	77.4%	82.6%
J48b	77.7%	76.7%
PART	71.1%	77.5%
Dtable	60.8%	78.0%
NB	55.4%	63.8%
IBk	49.0%	66.3%
baseline	48.0%	65.0%

The performance of symbolic induction methods is clearly better than that of other methods. In particular the decision tree induction algorithm (J48), with the use of multi-way splits has achieved the highest recall and precision results. Recall is slightly lower than precision, which is an indication that the trees are conservative in labelling instances as positive. This bias towards negative instances is due to the disproportionately large number of negative instances in the data set: 2,489 out of 3,516 instances.

One of the questions set initially was whether word-independent rules can be constructed, i.e., whether general patterns can be identified that are independent of specific words and still can be used for disambiguation. This question can be answered by examining the decision trees constructed by J48. In general, the constructed decision trees represent collections of word-specific disambiguators, i.e., they are composed of subtrees that focus on specific words. The only exception to this phenomenon is the use of the sense rank to group words that appear less frequently in the training set. For most of those words, the learning algorithms decide to select the most frequent sense (highest rank), rather than building complex disambiguation rules, using the collocates. This combination of general and word-specific disambiguation is desirable for large-vocabulary WSD.

When imposing the use of binary splits in the decision trees, the precision of the classifiers falls by 6 percentage points. The explanation for this fall in precision can be given by comparing the structure of the decision trees generated by J48b, with the structure of the trees generated by J48. As mentioned above, the trees generated by J48 perform primarily word-specific disambiguation, which means that the feature 'lemma' appears near the top of the tree, producing a broad tree. On the other hand, J48b is forced to produce binary trees, which have limited breadth. Each value of the

feature 'lemma', i.e., each ambiguous word, is treated as a separate feature. In this manner, it becomes difficult to build a word-specific disambiguator.

The rule induction algorithm PART suffers from a similar problem. Although it starts with a broad decision tree it breaks it into path-size segments and tries to optimise it by removing whole rules or parts of them. Each rule can only treat a single value of the feature 'lemma', which is equivalent to binarising the feature. However, it should be noted that the fall in performance observed for J48b and PART is accompanied by a much larger fall in the size of the disambiguation models. J48 generates trees in the order of a few thousand nodes (around 4,000 nodes), while J48b generates trees containing a few hundred nodes (around 850 nodes) and PART generates a few hundred rules (around 500). As a result, the models generated by J48b and particularly those generated by PART are much easier to understand than the trees generated by J48.

Outside the symbolic induction family, the algorithm that performs best is the one constructing decision tables. Its performance is not much lower than that of PART, which illustrates the fact that simple decision rules in the form of a table can be effective classifiers, approaching the performance of more expressive rule representations, such as that used by PART. At the same time, the large difference between this algorithm and IBk shows that the generalisation achieved by reducing the training set into a decision table can make a very important difference.

Finally, the Naïve Bayesian classifier and IBk perform very poorly. In particular, the performance of IBk is hardly above the baseline. The best result that we achieved for IBk was for $k=1$, i.e., adopting the class of the closest neighbour. The low performance of NB and IBk shows that generalisation is essential in order to arrive at an adequate solution to the WSD problem. This result may be affected significantly by our choice of representation for the data. In particular, one known problem with the instance-based classifiers is that they do not perform well when the instance space is sparse. This happens when there is a large number of features or feature values. In our representation we have used a small number of features. However, each feature can take many values. As a result, the instance space is very sparsely populated. In order to test the importance of this problem, we translated all of the multi-valued discrete features into a large number of binary features and selected a part of them, using an information theoretic measure (Information Gain). By doing that, in effect, we aimed to approach the representation used in J48b. Feature selection reduced considerably the sparseness of the instance space and improved the performance of NB and IBk. The results shown in Table 1 use the reduced binary feature set. Despite the improvement, the performance of the two algorithms has remained low, due to their inability to produce generalisations of the data.

6. Concluding Remarks and Further Work

Machine learning algorithms are a promising approach to the automatic construction of word sense disambiguators. In the study presented here we performed a

comparative evaluation of various supervised learning methods, which require that the training texts are hand-tagged with the correct senses for ambiguous words. Five learning algorithms were evaluated on financial news articles from the SEMCOR corpus. The textual data were translated into feature-vector training instances, as needed by the learning algorithms. 10-fold cross-validation was used to gain an unbiased estimate of the performance of the algorithms.

An important difference of the work presented here from previous work on this subject is the size of the vocabulary being disambiguated. Rather than restricting the attention of the system to a handful of words, all content words in the data were considered for disambiguation. This is a more realistic scenario, introducing the problem of sparseness of the training data. The reaction of the best-performing learning algorithm to this was to combine a simple general disambiguation filter for the words that appear less frequently in text, with word-specific disambiguation rules for the remaining words. This combination of word-specific and general disambiguation rules is an interesting outcome of our experiments that deserves further study.

The comparative evaluation has shown that inductive learning methods, which construct disambiguation models, by generalising the training data, perform best on the WSD task that we examined. In contrast, memory-based algorithms and a simple probabilistic one did not perform as well. This is attributed to the large number of values that each feature is allowed to take in the representation that we have used. This characteristic of the representation favours the algorithms that are able to perform flexible feature selection and ordering in the frame of an expressive model, such as a decision tree or a list of decision rules.

The above-mentioned observation presents an interesting direction for further work. Namely, the choice of an appropriate representation of the training data. In addition to the coding of the large number of feature values, we are concerned about the coding of the different senses for each word. The representation that was used here separates word instances into different senses, which are then treated as individual instances. Alternative representations that would allow the grouping of all senses related to a single word should also be examined.

Another important issue in WSD is the extent of the context used for disambiguation. Only local context was taken into account here. Topical evidence has also been shown to help in WSD and should be examined. Finally, we would like to test the learning methods in other sense disambiguation problems, in order to confirm the relative performance results that we presented here.

References

1. Black E.: An experiment in computational discrimination of English word senses. IBM Journal of Research and Development, v.32, n.2, (1988) 185-194
2. Bruce, R. and Guthrie, L.: Genus Disambiguation: A study in weighted preference. In Proceedings of the International Conference on Computational Linguistics, (1992) 1187-1191

3. Clark, P. and Niblett, T.: The CN2 algorithm. Machine Learning, 3(4), (1989) 261-283.
4. Cowie, J., Guthrie, J. A. and Guthrie, L.: Lexical disambiguation using simulated annealing. In Proceedings of the International Conference on Computational Linguistics, (1992) 359-365
5. Duda, R. O. and Hart, P. E.: Pattern Classification and Scene Analysis, John Wiley, (1973)
6. Gale, W. A., Church, K. W. and Yarowsky, D.: A method for disambiguating word senses in a large corpus. Computers and the Humanities, v.26, (1993) 415-439
7. Ide, N. and Veronís, J.: Introduction to the special issue on Word Sense Disambiguation: The state of the art. Computational Linguistics, v.24, n.1, (1998) 1-40
8. Kohavi R: The Power of Decision Tables. In Proceedings of the European Conference on Machine Learning, (1995) 174-189
9. Leacock, C., Towell, G. and Voorhees, E. M.: Corpus-based statistical sense resolution. In Proceedings of the ARPA Human Languages Technology Workshop (1993)
10. Leacock, C., Chodrow, M. and Miller, G. A.: Using corpus statistics and WordNet relations for sense identification. Computational Linguistics, v.24, n.1, (1998) 147-165
11. Lesk, M.: Automated sense disambiguation using machine-readable dictionaries: How to tell an pine cone from an ice cream cone. In Proceedings of the SIGDOC Conference, (1986) 24-26
12. Michalski, R. S., Mozetic, I., Hong, J. and Lavrac, N.: The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In Proceedings of the National Conference on Artificial Intelligence, (1986) 1041-1045.
13. Mitchell, T. M., Machine Learning, McGraw-Hill (1997)
14. Paliouras, G., Karkaletsis, V. and Spyropoulos, C. D.,:Machine Learning for Domain-Adaptive Word Sense Disambiguation. In Proceedings of the Workshop on Adapting Lexical and Corpus Resources to Sublanguages and Applications, International Conference on Language Resources and Evaluation, Granada, Spain, May 26 (1998)
15. Paliouras, G., Karkaletsis V. and Spyropoulos, C. D.: Learning Rules for Large Vocabulary Word Sense Disambiguation. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '99), v. 2, 674-679 (1999)
16. Quinlan, J. R.: C4.5: Programs for machine learning, Morgan-Kaufmann (1993)
17. Schütze, H.: Automatic word sense discrimination. Computational Linguistics, v.24, n.1, (1998) 97-124
18. Towell, G. and Voorhees, E. M.: Disambiguating highly ambiguous words. Computational Linguistics, v.24, n.1, (1998) 125-146
19. Wilks, Y. A., Fass, D. C., Guo, C. M., MacDonald, J. E., Plate, T. and Slator, B. M.: Providing machine tractable dictionary tools. Machine Translation, v.5, (1990) 99-154
20. Witten, I.H. and Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan-Kaufmann (1999)
21. Yarowsky, D.: Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In Proceedings of the International Conference in Computational Linguistics, (1992) 454-460
22. Yarowsky, D.: Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, (1994) 88-95
23. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, (1995) 189-196