Using Hierarchical Clustering to Enhance Classification Accuracy

Matthaios Theodorakis[1], Andreas Vlachos[2], Theodore Z. Kalamboukis[2]

[1] Department of Computing, Imperial College London, UK
matthaios.theodorakis@alumni.imperial.ac.uk
[2] Department of Informatics, Athens University of Economics and Business
{avl, tzk}@aueb.gr

Abstract. A new approach to classification is presented based on COBWEB, an unsupervised conceptual clustering algorithm. The modifications proposed improved the classification accuracy by 2.32% and up to 7.25% in the Period Disambiguation system that was built in order to test the efficiency of the approach. The system can be trained across different domains and languages. It has been tested on the Brown Corpus and on a collection of articles from Greek financial newspapers achieving accuracy 99.18% and 99.35% respectively.

1 Introduction

Classification is one of the most important issues in machine learning. The methods used to construct classifier have to be as flexible as possible in order to adapt to various problems and contexts with minimal effort. Therefore, methods that are trainable are more interesting than those that are tailored to the needs of a specific task. Several algorithms have been thoroughly studied in the literature and extensively used by many researchers. To mention a few of the most popular approaches: C4.5 algorithm for decision tree induction, [14], neural networks [12] and Bayesian classifiers [3, 9], transformation-based learning theory [2] and maximum entropy [15] have been proposed to deal with the problem of classification.

In this paper, a modified version of COBWEB [5], a well-known hierarchical clustering algorithm, is proposed and its performance is demonstrated in the period disambiguation problem. The period sign (".") can be used as part of numerical expressions, abbreviation marks and end-of-sentence marks. Sentence boundary detection has received attention in the past decade, as it is a task involved in many NLP applications. Named entity recognition, part of speech tagging, question answering, etc. all require sentence splitting at an early stage. Rule-based systems such as the Alembic system [1] and lexicon-based systems [10] use a large number of rules and resources such as lexicons and abbreviation lists. Obviously these approaches are language or even corpus specific and they receive a lot of criticism by other researchers ([11], [13] and others), because they lack portability and in order to achieve high results they need a lot of human assistance, Therefore, machine learning approaches have been proposed.

Riley in [16] using regression trees and a training corpus of 25 million words from a corpus of AP newswire achieved 99.8% accuracy in the Brown Corpus. Mikheev in [13] has presented a method that integrates part-of-speech tagging with sentence boundary detection. His system, trained on the Wall Street Journal Corpus and tested on the Brown Corpus demonstrated an accuracy of 99.8%. Even though this method does not need labelled data, it requires part-of-speech information, which limits its portability to other languages. Another approach that performs sentence boundary disambiguation using the maximum entropy model proposed by Reynar and Ratnaparkhi [15], achieved in the portable version 97.5% on the Brown Corpus and 98.0% on WSJ, when trained on 39,441 sentences of the latter.

In this paper, in order to exhibit the portability of conceptual clustering as a classification method used in period disambiguation, experiments were made with a Greek corpus too. The main obstacle in less studied languages like Greek is that there are no standard corpuses or other resources, such as lexicons, available. Therefore, such efforts are difficult to be accomplished and their results cannot be easily compared. For the Greek language, results reported by Stamatatos et al in [18] achieved 99.4% accuracy on Greek newspaper articles using a variation of the basic idea of the transformation-based learning [2] in order to automatically extract disambiguation rules from a training corpus. While this method does not need other resources than a pre-marked corpus, it has the disadvantage that the inclusion criteria for the rules produced were acquired empirically. Another approach by Schmid, [17], uses a Bayesian classifier that achieves high accuracy (>99.5%) in the Brown corpus and it is reported to be equally effective in German too. Even though, it has a great advantage that it does not require labelled data, in order to achieve optimal results it uses heuristics and optimizations that are language or corpus specific.

The rest of the paper is organised as follows: in section 2, the algorithm used – COBWEB, proposed by [5] – is briefly described and the approach is presented. In section 3, the modifications and additions to the algorithm are introduced that improve its performance. In section, 4 the performance of the algorithm is presented with experimental results from two corpuses, an English and Greek and finally, conclusions are made and future work is proposed.

2 Conceptual clustering in machine learning

Conceptual clustering is normally used in order to discover classes of objects with common characteristics in large amounts of data. A system employed in this task receives as input a set of observations and outputs a set of classes in which the input is distributed. The observations are described by a predefined set of attributes that take a value from a given set of values. The attributes are chosen such that to represent the observation's characteristics and assist in this way in forming a meaningful clustering scheme. Such a scheme produces a set of classes that exhibit internal similarity and external dissimilarity among them. Conceptual clustering systems evaluate these properties using an appropriate objective function and attempt to improve the quality of the clustering by employing a control strategy. The clusters produced are characterized by the attribute values of their observations. The quality of the clustering depends on whether the attribute values are predictive of and predictable by each cluster. If an observation is described by a value frequent in a class then it is highly probable that it belongs to that class (value predictive of the class). Conversely, if an observation is assigned in a certain class, then it is highly probable that the attribute values of the observation appear frequently among the attribute values of the class (value predictable of the class).

In this study the COBWEB clustering algorithm [5] is used. COBWEB is an incremental and unsupervised clustering algorithm that produces a hierarchy of classes. Its incremental nature allows clustering of new data to be made without having to repeat the clustering already made. The control strategy of COBWEB is described in the following pseudocode:

```
Function **Cobweb** (object, root)
  Incorporate object into the root cluster;
    If root is a leaf then
      return expanded leaf with the object;
    else choose the operator that results in the
    best clustering:
      a) Incorporate the object into the best host;
      b) Create a new class containing the object;
      c) Merge the two best hosts;
      d) Split the best host;
    If (a) or (c) or (d) then
      call Cobweb (observation, best host);
```

The objective function used to evaluate the quality of clustering produced by each of the above operators is called Category Utility (CU) and it is defined in [4], by:

$$CU(p) = \frac{\sum_{k=1}^{N_p} [\sum_i \sum_j P(C_k)\{P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2\}]}{N_p} \quad \textbf{(1)}$$

In this paper, modifications of COBWEB are examined in order to adapt the algorithm to the needs of the problem of classification. Thus COBWEB is used to produce a clustering from a set of training data with labelled observations and then is used to classify new unseen observations. During classification, each unseen observation is inserted into the “training” COBWEB tree without altering its structure. It becomes obvious that if an unseen observation is classified into one of the existing classes, then some of the attributes values can be inferred using information from the class whereto the observation has been classified. Thus when the observation reaches a leaf node, it is removed from the tree and it is given the most common label amongst the labels of the observations of that node.

3 Additions – Modifications

3.1 Pure Insert

Ideally, a clustering algorithm should assign only the observations that share common labels into the same cluster. However, the attributes of the observations do not provide enough evidence to the algorithm in order to discriminate between them and consider them in different classes. Thus, the algorithm constructs an inappropriate clustering scheme particularly due to misleading initial observations. In the classification problem though, the training data set is pre-marked and the labels of the observations are known. Thus to take advantage of the labels of the training set, a modification was made to the control strategy of COBWEB, in order to create more appropriate clusters that enhance the performance in the classification task.

Each time an observation is inserted into the clustering, it is dealt using the four operators (incorporate in the best host, create a new class, merge the two best hosts and split the best host). The so called, "pure insert", modification alters the "incorporate in the best host" operator and forces the algorithm to compare the label of the new observation with the labels of the observations already in the cluster. If they are the same then the observation is added in the host node as in the original version of the algorithm. If they are different, the observation is added, and the node obtains two children; one contains the observations that existed in the cluster-node before and the other contains only the new one. In this way, the leaf-nodes produced contain observations that share the same label. By comparing the labels, the algorithm is provided with a hint to create a clustering more appropriate. Consequently, a mixture of unsupervised and supervised clustering is achieved.

It might be argued that "pure insert" does not comply with the unsupervised nature of COBWEB. Certainly this feature is important in clustering when trying to discover classes in data but in classification, the classes are (commonly) known in advance. Therefore, although, COBWEB looses its main characteristic as a clustering algorithm, its use for categorisation aims to construct subclasses that effectively predicts the value of a certain attribute, instead of a more general classification scheme that could be created in a clustering task.

The disadvantage of the modified algorithm is that it produces a larger training tree that may slow down the time of clustering the training data. This is not however, a problem in categorisation since the training data set usually is not very large.

In our experiments, the benefit from "pure insert" varied between 3% and 15%, depending on the language and the size of the training set. More comprehensive results appear in Fig. 1 and Fig. 5 in section 4.

3.2 Pruning

Pruning is a procedure used to cut off singleton nodes that have been created during clustering. Pruning was mentioned by Fisher in [6], as a clustering optimisation technique, even though it is not applied in the same way here. The motivation to use pruning was that, as the experiments showed, most of the errors occurred when classifying an unseen observation into a singleton node. Most of the singleton nodes are created by the "pure insert" modification proposed earlier. These nodes may represent rare cases in the training data set or even errors. It is also possible that they have been created erroneously by the algorithm and therefore not used to contain other observations. This occurs usually in the beginning of the training. However, in our approach clustering is used to construct a classification scheme for unseen observations, not to cluster the observations themselves. Therefore, the removal of singleton nodes along with the observations contained in them is acceptable.

The results show that even though new errors may occur with pruning in comparison to the errors made without it, the overall performance of the classification task is improved. In our experiments, pruning combined with "pure insert" increased the accuracy by 0.5%, depending on the language and the size of the training set. It is worth noticing that when used without pure insert pruning does not affect the performance. Diagrams demonstrating the effect of pruning appear in section 4 (Fig. 2 and Fig. 6).

3.3 Retraining

Retraining is the procedure in which the training set is fed to the algorithm more than once. The order in which the observations are inserted is preserved. Retraining can be repeated as many times as desired, taking advantage of the incremental nature of the algorithm. This technique can be viewed as a way to decrease the "ordering effect" of COBWEB, as reported by Fisher et al. in [7]. Fisher, [6], proposes the iterative redistribution procedure as a way to improve the quality of the clustering. Retraining is inspired by this technique.

Retraining, as proposed in this paper, has the advantage that no special strategy is required in order to perform it. The same observations are inserted in the clustering one more time. From a machine learning point of view, retraining allows the algorithm to find better classification schemes, reviewing the same information. It is therefore a way to make better use of the training set available. The main disadvantage is that it may increase the size of the training tree and the computational cost of the learning phase.

The training tree produced achieved higher accuracy in sentence disambiguation, compared to the results obtained with a single training, as it can be observed in the diagrams in section 4 (Fig. 3 and Fig. 7). The benefit from this modification varies from 0.5% to 10% depending on the corpus and the size of the training set. This is a hint that there is knowledge in the training set that is not revealed with one train. Empirically, it has been observed that the 4th train does not improve the performance. Finally, as with pruning, retraining is effective only when coupled with pure insert.

3.4 Classification complexity

One of the basic requirements in clustering-categorization algorithms is the fast processing of new observations. The need for speed is obvious once we consider the on-line nature of the task. This is especially apparent in the problem of period disambiguation, which may be embedded in another on-line natural language processing application. Classification in our approach is accomplished by “inserting” the unseen observation into the training tree using the category utility function, without altering the training tree itself. The options (b), (c) and (d) of the algorithm are not considered in the classification phase.

In COBWEB’s algorithm the selection of the best operator for a node C_p depends on the maximum value of the CU function for all the nodes-children of C_p. Thus, it is not necessary to calculate the absolute values of CU and an equivalent simplified formula can be used instead. Since the COBWEB tree remains fixed during the testing phase, the new formula of CU exploits the data gathered in the nodes of the tree and takes into account only the information of the new incoming observation.

Consequently instead of the original CU function, it can be easily verified, that an equivalent simplified function, F(k), can be used, defined by:

$$F(k) = p(C_k)\sum_i\sum_j\{p(A_i = V_{ij} \mid C_k^{new})^2 - p(A_i = V_{ij} \mid C_k)^2\} - p(C_k)\sum_i\sum_j\{p(A_i = V_{ij}, C_k^{new})^2 - p(A_i = V_{ij}, C_k)^2\}\ , \quad \textbf{(2)}$$

where the first double sum denotes the difference of the intra cluster “density” with (C_k^{new}) and without (C_k) the new observation and the second the overall difference of the inter cluster “density” respectively. Substituting probabilities with integer counts relation (2) becomes:

$$F(k) = A_k + \frac{2}{c_k + 1} \times \sum_j c(A_i = V_{ij}, C_k) - \frac{1}{(c_p + 1)^2}\sum_j c(A_i = V_{ij}, C_p)\ , \quad \textbf{(3)}$$

where the factor $A_{k,}$ for each node k, represents a part of the utility function of the training COBWEB tree, j is referring to the attribute values V_{ij} of the new observation and c_k is the number of observations in node k. A_k is calculated by the end of the learning phase, and only the information of a new incoming observation are taken into account in each decision. Formula 3 has improved dramatically the execution time of the classification phase.

4 Experimental Results

To demonstrate the performance of the proposed modified COBWEB it has been applied to the problem of period disambiguation. For this purpose a training set is used where the periods of the texts are considered as the observations. It is worth mentioning that no dictionary is used, either for abbreviations or for ordinary words.

No other information such as part of speech tagging or specific domain knowledge is taken into consideration. This fact makes the solution to the period disambiguation problem presented here portable to various domains and different languages, achieving satisfactory results. The performance of the algorithm is measured by calculating the error-rate in the results. The error-rate is defined as the percentage of misclassified periods against all the periods in the testing set.

Conceptual clustering in period disambiguation has been tested on two corpuses, the Brown Corpus for the English language and a collection of articles taken from Greek financial newspapers. The same tests were performed on both collections in order to show the portability of the approach. The attributes for both languages are based on the word before, the word after, the length of the sentence and the classification of the previous period.

Each corpus was equally divided into training and testing set. Experiments were made using various portions (compared to the testing set) of the training set (1%, 5%, 10%, 20%, 30%, 50% and 100%). The testing set was the same for all the experiments (equal to the half of the collection).

4.1 The Brown corpus

The Brown Corpus [8] contains 15 large documents that vary from newspaper articles to literature. In order to create as representative sets as possible each document was equally split into training and testing. Each training set was constructed by taking the same percentage of sentences from each of the 15 documents in the corpus. The periods were selected from each document in the order they appear, without any other criteria. The fact that the articles of the Brown Corpus are taken from various sources (newspapers, literature, science, etc.) combined with the "ordering effect" that affects COBWEB performance makes the order in which the documents are fed to the algorithm important. The overall percentage of periods that are indeed end of sentences among all periods is 90%.

4.2 The Greek corpus

The Greek corpus used in our experiments is a compilation of articles from Greek financial newspapers retrieved from the World Wide Web containing 690 documents with a total of 13,577 periods. The separation of the corpus in training and testing sets was the same with the procedure followed for the English documents. The articles contain many abbreviations and periods that do not terminate sentences (38.59%). The attributes used are the same with those used for the English language with the addition of the attribute whether the last letter of the word before is consonant.

The experiments conducted include various combinations of the modifications proposed in the previous section. In the figures presented here (Fig. 4 and Fig. 8) only the best performing combinations are reported. The best training method, which, consistently performed best without fluctuations for all the experiments was obtained with 3 trains and then pruning. It is also important to notice that in this approach, training with a small amount of randomly selected data can achieve less than 1% error rate. Testing, using the modification proposed, lasts less than a minute for both languages (≈27,000 periods for the Brown corpus), even though the exact amount of time depends on the size of the training tree.

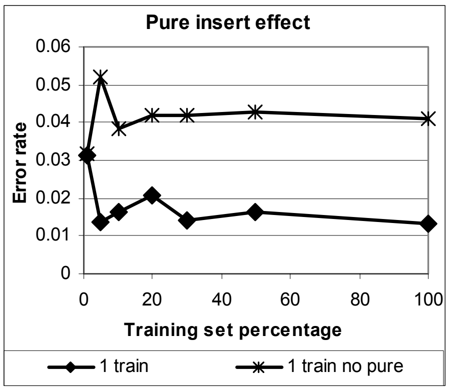

Fig. 1. Brown Corpus – Pure Insert.

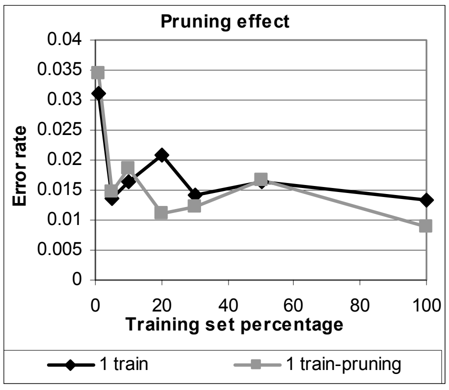

Fig. 2. Brown Corpus – Pruning Effect.

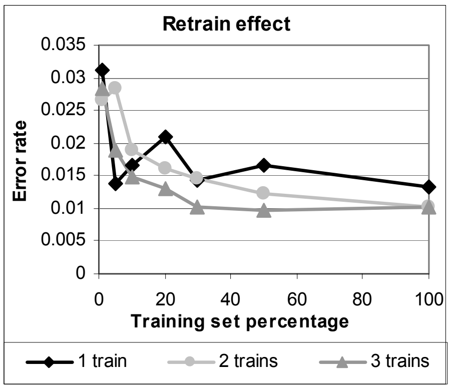

Fig. 3. Brown Corpus – Retrain Effect.

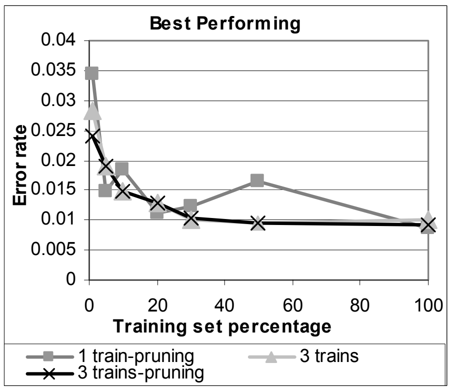

Fig. 4. Brown Corpus – Best Performing.

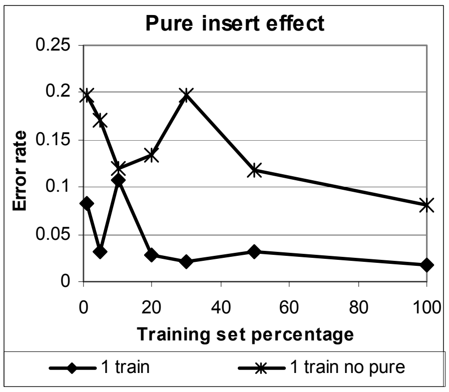

Fig. 5. Greek Corpus – Pure Insert.

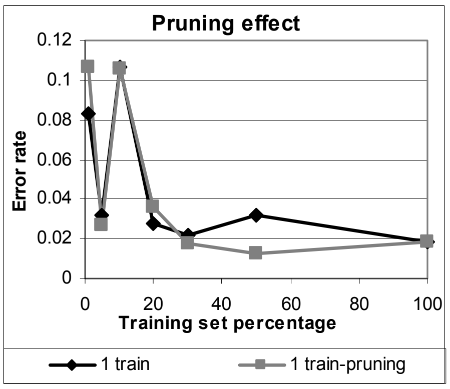

Fig. 6. Greek Corpus – Pruning Effect.

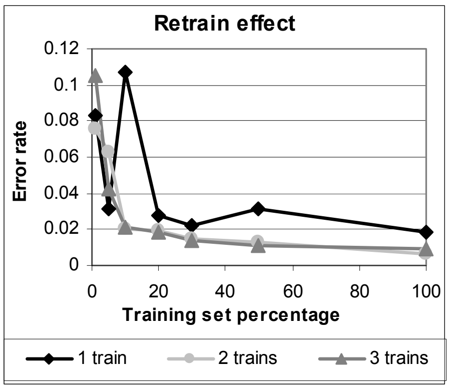

Fig. 7. Greek Corpus – Retrain Effect.

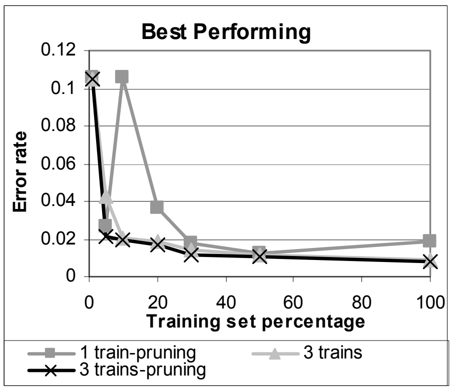

Fig. 8. Greek Corpus – Best Performing.

5 Conclusions

COBWEB, using the modifications proposed in this paper, achieved high accuracy in period disambiguation. Even though better results have been reported in the literature, as those reported by Mikheev [13], it is highly portable and requires very limited resources; it needs only a training set with pre-marked end-of-sentence candidate punctuation marks. It is worth mentioning that the modifications proposed improved the classification accuracy of COBWEB by 2.32% in the English corpus and up to 7.25% in the Greek language.

The attributes used in the experiments are very simple and applicable to any language. Another advantage is that it exhibits low error rate even with very small training sets. With a training of 1,382 periods from the Brown corpus (5% the size of the testing set) it achieves 1.8% error rate. This result can be compared to those reported in [15], who achieve 2.7% error rate using a maximum entropy model and 2,000 training sentences from WSJ, when testing it on the same corpus.

The main disadvantage of our approach is the computational cost during the training phase. For large training sets or when re-training is used, the calculations made in order to compute the value of the Category Utility function are expensive in time. Considering though the cost of compiling lexicons of abbreviations or part-of-speech information, it makes it worthwhile since these costs for other approaches are inevitable when switching domains or language. The second problem is the "ordering effect," which affects the training tree produced and consequently the results. Using the algorithm for classification it seems, from the experimental results, that the introduction of the pure insert procedure is a way to solve this issue.

Despite the disadvantages mentioned, conceptual clustering performed very well in period disambiguation in two languages without sophisticated knowledge. Most importantly, conceptual clustering itself is a quite generic classification method and probably can be applied to other disambiguation problems in NLP. Currently the performance of the algorithm for named entity recognition is under investigation together with a procedure for selecting automatically the attributes of the observations for a given problem.

References

1. Aberdeen, J., Burger, J., Day, D., Hirschmann, L., Robinson, P., Vilain, M.: Description of the alembic system used for muc-6. In: The Proceedings of the Sixth Message Understanding Conference (MUC-6). Morgan Kaufmann (1995).
2. Brill, E.: Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. Computational Linguistics, 21(4), (1995) 543-565.
3. Duda, R.O., Hart, P.E.: Pattern, Classification and Scene Analysis. New York: John Wiley & Sons (1973).
4. Gluck, M.A., Corter, J.E.: Information, uncertainty, and the utility of categories. Proceedings of the Seventh Annual Conference of the Cognitive Science Society. CA: Lawrence Erlbaum Associates, Irvine (1985) 238-287.
5. Fisher, D.H.: Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2, (1987) 139-172.
6. Fisher, D.H.: Iterative Optimization and Simplification of Hierarchical Clusterings. Journal of Artificial Research 4 (1996) 147-179.
7. Fisher, D.H., Xu, L., Zard, N.: Ordering effects in clustering. In Proceedings of the Ninth International Conference on Machine Learning. CA: Morgan Kauffmann, San Mateo (1992) 163-168.
8. Francis, S., Kucera, H.: Computing Analysis of Present-day American English. Brown University Press, Providence, RI (1967).
9. Friedman, N., Geiger, N., Goldszmidt, M.: Bayesian Network Classifiers. Kluwer Academic Publishers, Boston (1997).
10. Grefenstette, G., Tapanainen, P.: What is a word, what is a sentence? Problems of tokenization. In Proceedings of the 3rd Conference on Computational Lexicography and Text Research (COMPLEX ’94), Budapest (1994).
11. Kiss, T., Strunk, J.: Viewing sentence boundary detection as collocation identification. In Busesmannm S. (Hsrg.), Konvens 2002.
12. Lampinen, J., Laaksonen, J., Oja, E.: Neural Network Systems, Techniques and Applications in Pattern Recognition. Helsinki University of Technology, Department of Electrical Engineering, Laboratory of Computational Engineering (1997).
13. Mikheev, A.: Tagging sentence boundaries. In Proceedings of NAACL’2000, Seattle, (2000) 264-271.
14. Quinlan, J.R.: C4.5: Programs for Machine Learning. CA: Morgan Kaufmann, San Mateo (1993).
15. Reynar, J.C., Ratnaparkhi, A.: A maximum entropy approach to identifying sentence boundaries. In Proceedings of the Fifth Conference on Applied Natural Language Processing, Washington, D.C. (1997).
16. Riley M.D.: Some applications of tree-based modelling to speech and language indexing. Proceedings of the DARPA Speech and Natural Language Workshop (1989) 339-352.
17. Schmid, H.: Unsupervised Learning of Period Disambiguation for Tokenisation. Internal Report, IMS, University of Stuttgart (May 2000).
18. Stamatatos, E., Fakotakis, N., Kokkinakis, G.: Automatic Extraction of Rules for Sentence Boundary Disambiguation. In Proceedings of Workshop on Machine Learning in Human Language Technology, Chania, Greece (1999) 88-92.