Preventing Overfitting in Learning Text Patterns for Document Categorization

Markus Junker and Andreas Dengel

German Research Center for Artificial Intelligence (DFKI) GmbH, P.O. 2080, D-67608 Kaiserslautern, Germany

Abstract. There is an increasing interest in categorizing texts using learning algorithms. While the majority of approaches rely on learning linear classifiers, there is also some interest in describing document categories by text patterns. We introduce a model for learning patterns for text categorization (the LPT-model) that does not rely on an attribute-value representation of documents but represents documents essentially "as they are". Based on the LPT-model, we focus on learning patterns within a relatively simple pattern language. We compare different search heuristics and pruning methods known from various symbolic rule learners on a set of representative text categorization problems. The best results were obtained using the m-estimate as search heuristics combined with the likelihood-ratio-statics for pruning. Even better results can be obtained, when replacing the likelihood-ratio-statics by a new measure for pruning; this we call l-measure. In contrast to conventional measures for pruning, the l-measure takes into account properties of the search space.

Key Words: Text Categorization, Rule Learning, Overfitting, Pruning, Application

1 Introduction

Assigning text documents to content specific categories is an important task in document analysis. In office automation, systems for document categorization are used to categorize documents into categories such as invoices, confirmation of order [4]. These assignments can be used to distribute mail in the house, but they are also used for triggering document-specific information extraction tools, extracting, e.g, the total amount from an invoice. Automatic categorization systems rely on hand-crafted categorization rules of the form "**if** text pattern p_c occurs in document d **then** document d belongs to category c". Typical pattern languages rely on the boolean combination of tests on word occurrences. Within this language, the pattern (**or** (**and** gold jewelry) (**and** silver jewelry)) can be used to find documents which deal with gold or silver jewelry. More elaborated pattern constructs generally also allow tests on word sequences and word properties. The most prominent example for a rule-based automatic categorization system is *TCS* [6].

Since hand-crafting categorization rules is labor intensive, there is interest in automatic document categorization based on example documents. The majority of these algorithms rely on linear classifiers [12]. Until now, only a few rule learners have been applied to text categorization [1,3]. Nevertheless, rule learners offer some practical advantages:

- they produce *very compact classifiers*,
- the classifiers are *easy to understand and to modify* by humans (if, e.g., manual fine tuning is needed), and
- the classifiers are *portable* in the sense that the classifier can be used to query nearly any IR search engine.

In the remainder of the paper we first introduce our model for learning patterns for text categorization (Section 2). Based on a set of text categorization problems (described in Section 3) we then investigate different search heuristics to avoid overfitted pattern (Section 4), standard pruning techniques (Section 5), and a new pruning method relying on a measure we call l-measure (Section 6). A summary is is given in Section 7.

2 The LPT-Mo del

A text categorization problem in our sense is characterized as follows: Given

- an (in general infinite) set $\mathcal{D}$, the *text domain*;
- a category $\mathcal{K} \subset \mathcal{D}$, the *target category*
- positives $\mathcal{B}^{\oplus}$ and negatives examples $\mathcal{B}^{\ominus}$ for $\mathcal{K} \subset \mathcal{D}$, i.e. $\mathcal{B}^{\oplus} \subseteq \mathcal{K}$ and $\mathcal{B}^{\ominus} \subseteq \mathcal{D} \setminus \mathcal{K}$
- a representation language T_{LPT} for documents and a transformation $t_{LPT} : \mathcal{D} \rightarrow T_{LPT}$
- a pattern language $\mathcal{P}$ with an associated function $m : \mathcal{P} \times T_{LPT} \rightarrow \{0, 1\}$

We search for an algorithm that computes a pattern $p_{\mathcal{K}} \in \mathcal{P}$ from $\mathcal{B}^{\oplus}$ and $\mathcal{B}^{\ominus}$ which approximates

$$m(p_{\mathcal{K}}, t_{LPT}(\delta)) = \begin{cases} 1 : \delta \in \mathcal{K} \\ 0 : \delta \in \mathcal{D} \setminus \mathcal{K} \end{cases}, \text{ the } \textit{match function.}$$

In our model, real world documents (usually given as character sequences) are transformed via the transformation function t_{LPT} into the LPT document representation language. Within this language a document d is represented as a word sequence $(w_1 \ldots w_n)$ while a word w_i is represented as a character sequence. It is important to note that the transformation preserves almost all information in the original document. This allows us to, e.g. define pattern language constructs that rely on word sequences or complex word properties.

We did not commit to a specific pattern language $\mathcal{P}$ but only give a general frame for simplifying the construction of pattern languages. Here, we only introduce one very simple instance of possible pattern languages. This instance relies on a set of so-called *word tests* with $w \in \mathcal{P}$ iff w is a word. The semantics of word tests is given by

$$m(w, (w_1, \ldots, w_n)) = \begin{cases} 1 : \exists_i w = w_i \\ 0 : else \end{cases}$$

Based on word tests we define the simple pattern language used in the following by $k_1 \vee k_2 \vee \cdots \vee k_n$ with $k_i = w_{i,1} \wedge \cdots \wedge w_{i,m_i}$ and $w_{i,j}$word test The LPT-document representation also allows much more complex pattern languages which rely on word orderings and word properties [7].

Learning Algorithm The search for patterns is done using a separate-and-conquer algorithm as used in the well-known rule learner *CN2* [2]. To simplify the representation we use $\oplus$ (respectively $\ominus$) as a unary operator that returns the set of all positives (respectively negatives) example documents for a given example document set. The inputs of the algorithm are example documents B for the target category, a search heuristics sh and a pruning method sig (figure 1). It first initializes the document set R by B and the pattern p by false. Using the function s the algorithm then chooses the "best" conjunction of words c_{best} for describing a subset of the positives examples in R. All documents of R covered by c_{best} are then removed from R and the algorithm iterates until the best conjunction of words results in true. The disjunction of all conjunctions given by s is the pattern p which is returned as learning result.

```
Input: Example documents $B$ for target category
Output: "best" pattern $p$ for target category
  $R \leftarrow B$, $p \leftarrow$ false
  repeat
    $S = s(B, R, \{\text{true}\}) \cup \{\text{true}\}$
    $c_{\text{best}} \leftarrow$ "best" conjunction in $R$
    $R \leftarrow R \setminus \{d \in R^{\oplus} \mid m(c_{\text{best}}, d) = 1\}$
    $p \leftarrow p \vee c_{\text{best}}$
  until ($c_{\text{best}}$ = true)
```

Fig. 1. Algorithm for finding the best pattern

Finding the best conjunction c_{best} is done using the algorithm shown in figure 2. The algorithm starts with the empty conjunction as the best conjunction c_{best}. Within a loop this conjunction is refined by word tests. From those refinements that are significant with respect to sig, the best one according to the search heuristics *sh* is taken as the new best conjunction. The loop ends if no better significant refinement than the current c_{best} can be found anymore.

3 Methods and Material

For the experimental evaluation we rely on four text collections with positive and negative examples for 98 different categories: a collection of German technical abstracts with categories such as "computer science and modeling"

```
Input: Example documents $R$, search heuristic sh, significance criterion sig
Output: "best" conjunction $c_{\text{best}}$ in $R$ for target category
  $c_{\text{best}} \leftarrow$ true, $c \leftarrow c_{\text{best}}$
  repeat
    $S = \{c_{\text{best}} \wedge w \mid w$ word occurring in$R\}$
    $S_{\text{pruned}} = \{c \in S \mid \text{sig}(B, R, c_{\text{best}}, c)\}$
    $c \leftarrow c' \in S_{\text{pruned}}$ with best value of $\text{sh}(R, c')$
  until $c_{\text{best}} = c$
```

Fig. 2. Algorithm for finding the best conjunction

and "classification, document analysis and recognition", a collection of office documents which consists of OCR'ed German business letters and with categories such as "invoice" and "offer" and two freely available English collections: the *Reuters*-collection[1] and a collection of newsgroups articles [2]. For the evaluation we split each collection 1:1 in a learning and a test set.

For the evaluation of learned patterns on the test set, we use the effectiveness measures recall and precision which are widespread in information retrieval [9]. Recall and precision correspond to the characteristic requirements on patterns: They should cover a category as completely as possible (measured by the recall) and they should cover it as correctly as possible (measured by the precision). For each learned pattern $p = k_1 \vee \cdots \vee k_n$ a range of recall/precision values corresponding to $p_i = k_1 \vee \cdots \vee k_i$ for $i = 0 \ldots n$ was computed. Averaging over the categories was done at predefined recall points by averaging the approximated precision for all patterns at these recall points. The resulting range of recall/precision points will be graphically shown in recall/precision diagrams. The little arrows in the diagrams shown later indicate the minimum increase in recall or effectiveness needed at certain recall/precision points for significance according to the p-Test as used in [12] (error probability 5%).

4 Search Heuristics

In literature in different rule learners various search heuristics for complexes are used [5] of which some of the most prominent ones are (we use the abbreviations $p = |m(k, R^{\oplus})|$, $n = |m(k, R^{\ominus})|$, $P = |R^{\oplus}|$, and $N = |R^{\ominus}|$): *Precision* ($\frac{p}{p+n}$), *Information Content* ($-\log(\frac{p}{p+n})$), *Entropy* ($-\log(\frac{p}{p+n})$), *Laplace Estimate* ($\frac{n+1}{n+p+2}$), *Accuracy* ($\frac{p+(N-n)}{P+N}$), and m-*Estimate* ($\frac{p+m\frac{P}{P+N}}{p+n+m}$). Figure 3 compares the effectiveness we obtained on our text categorization problems with the search heuristics. To alleviate problems arising with small coverages when using precision, information content and entropy the order

[1] available via `http://www.research.att.com/~lewis`

[2] available via `http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html`

given by these function were refined as follows: If two complexes obtain the same weight, the one which covered more documents was preferred (i.e. the one with $p + n$ maximal). In addition, for the entropy, complexes with $\frac{p}{p+n} < \frac{1}{2}$ were excluded. For the m-estimate we evaluated the parameters $m \in \{0.01, 1, 10, 20, 50, 100\}$.

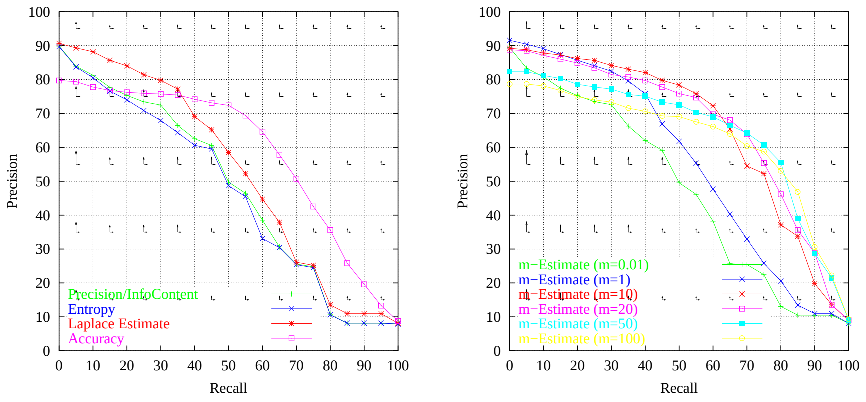

Fig. 3. Influence of the search heuristics on the effectiveness

It can be observed that the effectiveness of precision/information content and entropy do not differ much.This can be explained by the equivalence of precision/informations content and entropy for $\frac{p}{p+n} > \frac{1}{2}$. Compared to precision/information content and entropy the Laplace estimate performs better. This can be explained by the weighting of complexes with small coverages close to $\frac{1}{2}$. The rather pessimistic estimate gives, e.g., a better weight to a complex with $p = 100$ and $n = 1$ than for a complex with $p = 2$ and $n = 0$. The accuracy shows a lower precision on a lower recall, but obtains a higher precision at higher recalls than the other search heuristics. In general by going from precision/information content and entropy to the Laplace-estimate and from the Laplace-Estimate to the accuracy a high precision at low recalls is replaced by a high precision at higher recalls. The same also holds for the m-estimate with increasing m. Choosing the right value for m for each recall, a precision can be obtained that is better than the one of all other search heuristics on this recall.

5 Pruning Methods

Table 1 shows the pruning methods we have evaluated on our text categorization problems. The method *syntactic constraints* restricts conjunctions to a maximal number of arguments $n_{\max}$. The intention for this method is the belief that complexes with more arguments are more likely to obtain a good rating by the search heuristics just by chance. Another pruning method, *minimum coverages*, requires complexes to cover at least c positive examples in the remaining learning set R. It follows the belief that the values of the search heuristic are more reliable if more (positive) examples can be used for its computation. Furthermore, we considered two pruning methods relying on the likelihood ratio statics (lrs). This measure gives high significance to complexes whose distribution of covered positive and negative example differs much from the distribution on some reference set. The variants of the lrs differ in what set it used as reference set. In the *CN2* variant used in the rule learner *CN2* the whole example set B is used as reference, i.e. $P' = |B^{\oplus}|$ and $N' = |B^{\ominus}|$. The *BEXA* variant (used in *BEXA* [11]) uses the predecessor complex k_{pre} of the complex to be judged, i.e. $P' = |m(k_{\text{pre}}, R^{\oplus})|, N' = |m(k_{\text{pre}}, R^{\ominus})|$.

syntactic constraints	$\text{sig}(B, R, k_{\text{pre}}, w_1 \wedge \cdots \wedge w_n) = \begin{cases} 1 : n \leq n_{\max}, n_{\max} \in I\!N \\ 0 : \text{else} \end{cases}$
minimum coverages	$\text{sig}() = \begin{cases} 1 : \lvert m(k, R^{\oplus}) \rvert \geq c \\ 0 : \text{else} \end{cases}$
lrs	$\text{sig}() = \begin{cases} 1 : -2\left(p \log\left(\frac{p}{(p+n)\frac{P'}{P'+N'}}\right) + n \log\left(\frac{n}{(p+n)\frac{N'}{P'+N'}}\right)\right) \geq \vartheta \\ 0 : \text{else} \end{cases}$

Table 1. Pruning methods and their definitions

Figure 4 summarizes the results obtained with the different pruning methods (the diagram showed in the lower right will be subject of the next section). The results of the *CN2* and *BEXA* variants are practically indistinguishable. It can be observed that:

- The method *minimum coverages* (upper right) has practically no positive influence on the effectiveness over the whole recall range. Increasing the minimal number of documents in $R^{\oplus}$ even decreases the precision.
- The effect of the other pruning methods *syntactic constraints* (upper left) and *likelihood ratio statics* (lower left) depends on the respective parameter and is very similar. By varying the parameter, high precision at low recall levels is traded for high precision at high recall levels.
- Pruning with the *likelihood ratio statistics* (lower left) slightly outperforms *syntactic constraints* (upper left). While with the lrs and $\vartheta = 20$ the precision is higher at the high recall level, the corresponding decrease in precision at lower recall levers is less dramatic than when using *syntactic constraints*.

g

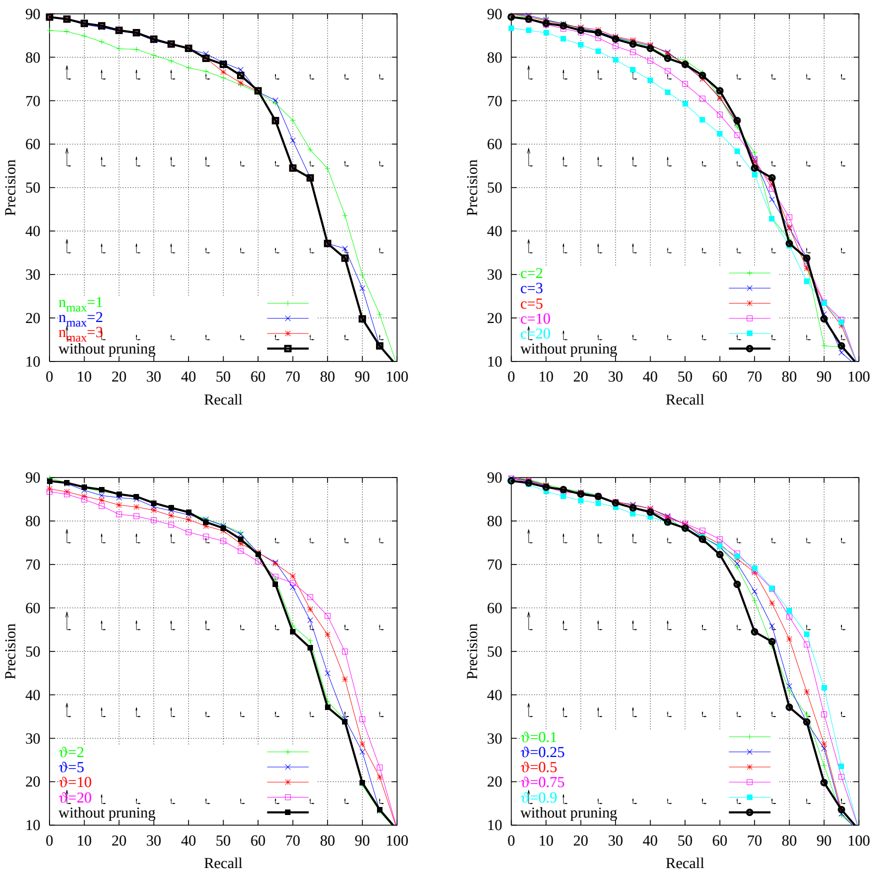

Fig. 4. Influence of pruning methods on the effectiveness: *syntactic constraints* (upper left), *minimum coverages* (upper right), *lrs* (*BEXA*) (lower left), *l-measure* (lower right)

6 A New Pruning Method

In this section we propose a new pruning method that does —in contrast to the methods listed in table 1— takes the size of the search space into account. In order to motivate the new significance measure, table 2 shows a part of the learning process for the category *invoice* in the German office document collection[3]. The first column contains the i-th iteration in the separate-and-conquer algorithms according to figure 1. For each iteration step the following two columns contain the number of remaining positives ($|R^{\oplus}|$) and negatives

[3] The words in the table translate as follows: rechnung(s) – invoice, zahlung – payment, angeben – specify

documents ($|R^{\ominus}|$). The following columns show the best conjunctions built within the algorithm to find the best conjunction (cmp. figure 2) as well as the number of positives and negatives examples $p = |m(c_{i,j}, R^{\oplus})|$ respectively $n = |m(c_{i,j}, R^{\ominus})|$ they cover in R. The column $\mathrm{sh}(R, c_{i,j})$ shows the value of the m-estimate for $c_{i,j}$ ($m = 10$). Finally, the last column contains the value $l = \left|\left\{c \in S \mid |\, m(c,R)| = |m(c_{i,j}, R)|\right\}\right|$ with $S = \{c_{i,j-1} \wedge w \mid w \in \mathrm{words}(R)\}$ the *local search space*. According to this definition l is the number of refinements of $c_{i,j-1}$ that covers the same number of documents in R as the conjunction $c_{i,j}$.

i	$\|R^{\oplus}\|$	$\|R^{\ominus}\|$	j	conjunction $c_{i,j}$	p	n	$\mathrm{sh}(R, c_{i,j})$	l
1	357	513	0	true	357	513	0.41	
			1	rechnung	203	54	0.78	2
			2	rechnung $\wedge$ rechnungs	37	1	0.87	5
2	320	513	0	true	320	513	0.38	
			1	rechnung	166	54	0.74	4
			2	rechnung $\wedge$ zahlung	57	2	0.78	3
			3	rechnung $\wedge$ zahlung $\wedge$ angeben	28	2	0.80	2
.	.	.	.	.	.	.	.	.
10	171	513	0	true	171	513	0.25	
			1	rechnung	60	54	0.50	5
			2	rechnung $\wedge$ 10060	8	0	0.58	64
.	.	.	.	.	.	.	.	.
19	119	513	0	true	119	513	0.19	
			1	1157	4	1	0.39	506

Table 2. Capture of the learning process for the category *Rechnung*

In order to motivate the l-measure we first consider the refinement step from $c_{19,0}$ to $c_{19,1}$. The conjunction $c_{19,0}$ covers 119 positives and 513 negatives examples. The probability of an example being a positive one can thus be estimated with $\frac{119}{119+513} \approx 0.19$. This probability can be interpreted as the a-priori probability for covering a positive example by a refinement of $c_{19,0}$.

The value of the m-estimate for $c_{19,0}$ is $\frac{4+10\frac{119}{119+513}}{4+1+10} \approx 0.39$. Each refinement of $c_{19,0}$ that covers exactly 5 documents (as does $c_{19,1}$) has a weighting at least as good as 0.39 iff the refinement covers at least as many positives examples as $c_{19,1}$: 4. The probability for covering randomly at least 4 positive examples can be computed using the a-priori probability estimated by $c_{19,0}$ with $\binom{5}{4}0.19^4(1-0.19)^1 + \binom{5}{5}0.19^5(1-0.19)^0 \approx 0.0055$ The low value indicates that the observed good coverage of $c_{19,1}$ has not occurred by chance. Thus, we would not prune $c_{19,1}$.

If we take into account how many refinements of $c_{19,0}$ exist that also cover exactly 5 documents, the result changes. This number is given as l in table 2 with 506. If we assume that the coverage of all 506 words that cover exactly 5 documents is random, the probability of covering 4 or more positive examples by chance (and thus the probability of getting a better weight than 0.39) increases drastically. It can be approximated by $\sum_{i=1}^{509} \binom{509}{i} 0.0055^i (1 - 0.0063)^{(509-i)} = 1 - (1 - 0.0055)^{509} \approx 0.94$. In this way, the consideration of the value l computed by the local search space would suggest we should prune $k_{19,1}$.

Without going into the details of the general derivation, the idea given by the above example can be used to formulate a new significance measure as follows. Using

$$\mathrm{Bin}\,(p+n, w, p) = \sum_{i=p}^{p+n} \binom{p+n}{i} w^i (1-w)^{p+n-i}$$

and

$$l = \left| \left\{ c' \in S \;\middle|\; |m(c', R)| = |m(c, R)| \right\} \right| ,$$

the new pruning method can be formulated as

$$\mathrm{sig}(B, R, c_{\mathrm{pre}}, S, k) = \begin{cases} 1 : \mathrm{sigm}(B, R, c_{\mathrm{pre}}, S, c) \geq \vartheta \\ 0 : \mathrm{else} \end{cases} \quad \text{with}$$

$$\mathrm{sigm}(B, R, c_{\mathrm{pre}}, S, c) = 1 - (1 - \mathrm{Bin}\,(p+n, w, p))^{\left|\left\{ c' \in S \middle| \; |m(c',R)| = |m(c,R)| \right\}\right|}$$

The results of the new pruning method using the l-measure are shown in the lower right corner of figure 4 for $\vartheta \in \{0.25, 0.5, 0.75, 0.9\}$. It can observed that with increasing ϑ the precision for higher recall values can be increased without harming the precision at lower recall as much as the likelihood-ratio-statistics does. Thus, the new pruning method — taking account of the local search space— outperforms the previously investigated pruning methods for our text categorization problems.

7 Summary

In this paper we have shown our model for learning patterns for text categorization. In contrast to the conventional models we do not use an attribute value representation of documents but represent documents as they are, i.e. as a sequence of words which again are represented by a sequence of words. This enables us to introduce complex pattern languages with patterns relying on word orders and word properties. Within the model, in this paper, we focused on preventing from overfitting in a simple pattern language, in which pattern are disjunctions of conjuncted word tests. We proposed and evaluated different methods for optimizing the effectiveness of patterns within this language. Of all search heuristics we evaluated the best results on our set of text categorization methods was obtained with the m-estimate. In addition to search heuristics we investigated pruning methods for text categorization. The best pruning method we found was the likelihood-ratio-statistics. We proposed a pruning method relying new measure we call l-measure. To our knowledge this is the only pruning method that takes into account the size of the search space. An experimental comparison of the new methods showed that the new pruning method gives better effectiveness than the likelihood-ratio-statistics.

References

1. C. Apté, F. Damerau and S. Weiss. Towards Language Independent Automated Learning of Text Categorization Models. In: *Proceedings of the 17th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval (SIGIR 94)*, page: 23–30, Dublin, Ireland, July 3-6 1994.
2. P. Clark and T. Niblett. The CN2 Algorithm. *Machine Learning*, 3(4) Seite: 261–283, 1989.
3. W.W. Cohen. Learning to Classify English Text with ILP Methods. In: *Advances in Inductive Logic Programming*, page: 124–143. IOS Press, 1996.
4. A. Dengel und K. Hinkelmann. The Specialist Board — A Technology Workbench for Document Analysis and Understanding. In: *Proceedings of the 2nd World Conference on Integrated Design and Process Technology (IDPT '96)*, page: 36–47, Austin, TX, USA, December 1996.
5. J. Fürnkranz. Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, 13(1) Seite: 3–54, 1999.
6. P.J. Hayes, P.M. Anderson, I.B. Nirenburg und L.M. Schmandt. TCS: A Shell for Content-Based Text Categorization. In: *Proceedings of 6th Conference on Artificial Intelligence Applications*, page: 320–326, Santa Barbara, CA, USA, May 5-9 1990.
7. M. Junker. *Heuristisches Lernen von Regeln für die Textkategorisierung.* Dissertation, University of Kaiserslautern, Germany, 2000 (in German).
8. J.R. Quinlan. Introduction of Decision Trees. *Machine Learning*, 3 Seite: 81–106, 1986.
9. C. van Rijsbergen. *Information Retrieval.* Butterworth, London, England, 1979.
10. C. Schaffer. Overfitting Avoidance as Bias. *Machine Learning*, 10(2) Seite: 233–241, February 1993.
11. H. Theron und I. Cloete. BEXA: A Covering Algorithm for Learning Propositional Concept Descriptions. *Machine Learning*, 24 Seite: 5–40, 1996.
12. Y. Yang und X. Liu. A Re-Examination of Text Categorization Methods. In: *Proceedings of the 22th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval (SIGIR 94)*, page: 42–49, Berkeley, CA, USA, August 15-19 1999.