Text Categorisation Using Document Profiling

Maximilien Sauban and Bernhard Pfahringer

Department of Computer Science
University of Waikato
Hamilton, New Zealand
{m.sauban,bernhard}@cs.waikato.ac.nz

Abstract. This paper presents an extension of prior work by Michael D. Lee on psychologically plausible text categorisation. Our approach utilises Lee's model as a pre-processing filter to generate a dense representation for a given text document (a document profile) and passes that on to an arbitrary standard propositional learning algorithm. Similarly to standard feature selection for text classification, the dimensionality of instances is drastically reduced this way, which in turn greatly lowers the computational load for the subsequent learning algorithm. The filter itself is very fast as well, as it basically is just an interesting variant of *Naive Bayes*. We present different variations of the filter and conduct an evaluation against the Reuters-21578 collection that shows performance comparable to previously published results on that collection, but at a lower computational cost.

1 Introduction

In the last decade the amount of textual information in digital form has grown exponentially, mainly due to the forever-increasing accessibility of the Internet. It is crucial to create tools to organise the amount of information available. Text categorisation is one such tool. It aims at classifying textual documents into predefined categories. Text categorisation applications are manifold and are ranging from *automated meta-data extraction* for indexing to *document organisation* for databases or web pages (see Yang et al., [1]). Other interesting uses of text categorisation include *text filtering*, generally as part of a producer-consumer relationship, or *word sense disambiguation* when dealing with natural languages processing (see Roth, [2]).

1.1 Existing Text Categorisation Methods

It is difficult to be exhaustive when listing the existing text categorisation methods. Amongst the main approaches, decision tree methods ("divide-and-conquer" approach) have the advantage of being "human readable" in the sense that they deal with symbolic entities and not numeric values [3]. Investigations using probabilistic models usually focus on *Naive Bayes* and its variants [4]. Joachims [5] introduced the support vector machine method to text categorisation. Also

N. Lavrač et al. (Eds.): PKDD 2003, LNAI 2838, pp. 411–422, 2003.

worth mentioning is the Rocchio method [6]; this method creates a prototype document for each class from the training set. A test document will be assigned to the class of the closest prototype found. Yang [7] invented a mapping approach using a multivariate regression model and investigated, together with Pedersen, lazy learning for text categorisation [8]. Frank et al. [9] investigated text categorisation using compression models, and Wiener et al. [10] were using neural networks. Yet other approaches have tried to improve predictive performance by incorporating semantic information like WordNet hypernyms [11].

1.2 David Lee's Method

Lee [12] came up with a psychologically plausible approach considering three different insights. Firstly, Lee noted that people are able not only to state that a given document is about a given topic but also that a document is not about a topic. Take for example "middle east conflict" as a topic; the occurrence of the word "rugby" in a document would give a strong hint about the document not being about the topic. Secondly, humans are able to make non-compensatory decisions: one can decide if a document is about a topic or not without necessarily having to read the whole document. Using our previous middle-east conflict example, if the document starts with something like "The south African rugby team just arrived in Auckland..." most people would not need to read any further to reach a conclusive, in this case negative, decision. Thirdly, people have the capacity to give answers with a level of confidence and so they are able to state if a document is either definitely about a topic or alternatively just remotely related to a topic.

Lee's model's formal definition is based on a Bayesian analysis, which states that it is possible to compute the posterior odds of a document being about a topic or not by multiplying the prior odds—chances of a document to be about a topic before looking at it—and the evidence—probability that a document would have been generated under the assumption that it is about a topic (or not). Lee considers the document as a *sequence* of words. The evidence then becomes the product of the probability of each word being in a document about the topic (or not). Note that this approach follows the *Naive Bayes* assumption that all words are independent of one another, also called the independence assumption. The evidences' probabilities are quantified using the number of occurrences of the given word over the total number of words and are calculated for both categories, for and against the topic. The independence assumption allows analysing words sequentially, which permits monitoring the evolution of the posterior odds word by word in the same order as they appear in the document. Considering the logarithm of the posterior odds and using evidences for and against the topic leads to the following equation:

$$\ln \frac{\Pr\,(c_j|d_i)}{\Pr\,(\neg c_j|d_i)} = \ln \frac{\Pr\,(c_j)}{\Pr\,(\neg c_j)} + \sum_{k=1}^{n} \ln \frac{\Pr\,(w_{ni}|c_j)}{\Pr\,(w_{ni}|\neg c_j)}$$

Figure 1 shows the evolution of the posterior odds of two documents processed by the text classifier. The graph on the left depicts the partial log-odds sums

for a document about the topic, while the graph on the right depicts those sums for a document that is not about the topic. Note that the partial log-odds sums are computed over larger and larger initial subsequences of the document, which causes the order of the words in the document to become significant. This example also shows the possibility of non-compensatory decision making by setting two thresholds, one for the document being about the topic and the other for the document not being about the topic. The decision is taken when one of the thresholds is reached. Let's assume the thresholds in Figure 1 are 100 for a document about the category and -20 for a document not about the category. In the left hand side case, the decision is taken after reading the 120^{th} word (when the curve meets with $y = 100$). On the right hand side example, the decision can be taken after reading the 45^{th} word (the curve meets with $y = -20$).

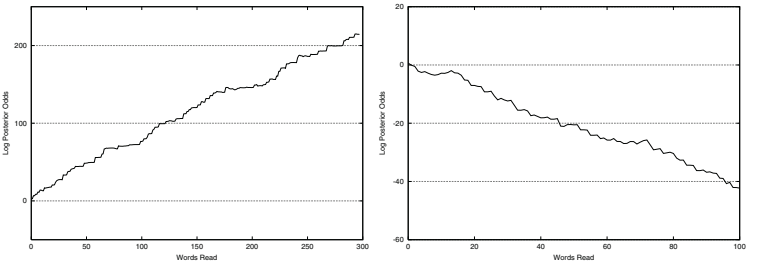

Fig. 1. Illustration of document profiles, the left hand side one is about the topic while the right hand side one is not.

1.3 Our Approach

The investigation presented here is an extension of Lee's work [12]. An interesting aspect of Lee's method is that the document is processed sequentially and the odds of the document with respect to a given category can be tracked as the words are fed to the system. We call this sequence of the partial sums of the log-odds of the words of a document a *document profile*. Usually those profiles are not as clear-cut and easy to classify as the ones shown in Figure 1. We have therefore decided to investigate a two-step process, where a first step generates document profiles according to Lee's method, and a second step extracts propositional information from these profiles that then can be fed into any arbitrary propositional learner. Thus, Lee's system is used as a dimensionality-reducing pre-processing step.

The next section will explain this process in more detail, discuss issues with the dictionary, and basically describe two different ways of extracting attributes from document profiles. Sections 3 and 4 explain the experimental setup and give and discuss experimental results. In Section 5 we present conclusions and discuss further work.

2 Generating and Manipulating Document Profiles

To construct a model, each word in the vocabulary is assigned two probabilities, the probability of the word being about the topic Pr $(w_k|c_j)$ and the probability of the word not being about the topic Pr $(w_k|\neg c_j)$ where w_k is the word, and c_j the topic. The word's *influence* $(\mathcal{I}_{w_k})$ is then calculated as follow:

$$\mathcal{I}_{w_k} = \ln \frac{\Pr\ (w_k|c_j)}{\Pr\ (w_k|\neg c_j)}$$

The probabilities are based on the rate at which the word has occurred in the training documents about and not about the category. Figure 2 (on the left hand side) portraits one such dictionary. The higher the magnitude of the influence, the more weight the word will have in the final decision. Note that there are much more words with a negative influence. The explanation for that lies in the skewedness of the training data: the dictionary pictured in Figure 2 (on the left hand side) was trained with 197 documents about a given category and 9,406 documents not about that category. The 9,406 negative examples used for training were in fact the union of all 89 other categories. The skewedness also explains why the maximum positive amplitude is greater than the maximum negative one. Specialised words of the category in focus (word with a large positive influence score) are more likely to have a denser concentration in the positive documents than the specialised words of the other category (actually categories).

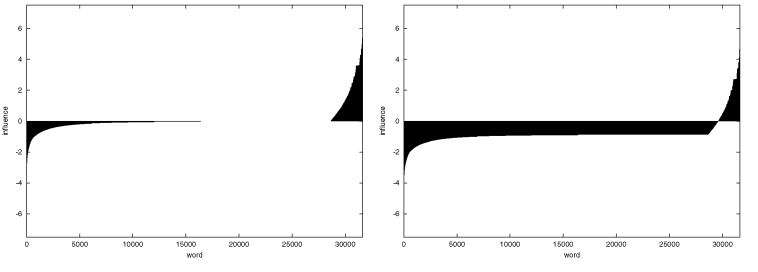

Fig. 2. Dictionary for one category of the Reuters dataset and a shifted version of the dictionary on the right.

The three following sub-sections will describe dictionary manipulations that proved to be beneficial, and explain the two ways propositional attributes are extracted from document profiles.

2.1 Shifting the Origin on the y-Axis

Figure 2 (right hand side) illustrates the result of shifting the origin on the y-axis in an attempt to equalise the maximum amplitudes. To shift the dictionary,

we subtracted half the sum of the top positive value and the top negative value from all the words present in the dictionary or more formally:

$$\forall k \in d : \mathcal{I}'_{w_k} = \mathcal{I}_{w_k} - \frac{max(\mathcal{I}_w) + min(\mathcal{I}_w)}{2}$$

Where $\mathcal{I}_{w_k}$ and $\mathcal{I}'_{w_k}$ are respectively the influence of the word before and after shifting, $max(\mathcal{I}_w)$ the largest influence in the dictionary and $min(\mathcal{I}_w)$ the smallest. Note that a lot more words now have a negative influence, and that the magnitude of the positive influences has been reduced. This shift usually improves performance. A more sophisticated threshold selection method might fare even better.

Table 1. Dictionary sizes after cutting off at x% of the top influence value.

percent of the maximum positive/negative value cut off	**# of words remaining in the dictionary**
0% (whole dictionary)	31651
10%	4046
15%	2860
30%	1288
60%	166

2.2 Reducing the Size of the Dictionary

As mentioned earlier, the specialised words carry a large influence score, but their distribution is highly skewed: there are almost no specialised words for the negative class. On the other hand, words with a low influence score, are more evenly distributed between both the positive and negative class. Their low influence score causes them to play only a minor part in the final decision, but they can potentially add noise. We have therefore introduced a mechanism to prune words from the dictionary based on their influence score. The decision threshold is based on the maximum positive value and the maximum negative value (of the unshifted dictionary). The cut off value is determined as a percentage of the maximum values. A cut at 30% means that all the words with influence score between 0 and 30% of the maximum positive influence and between 0 and 30% of the maximum negative influence score will not be taken into account. Table 1 shows the non linear relation between the cut off value and the number of words left in the dictionary when applying this idea to the dictionary of Figure 2. This pruning effect is also illustrated in Figure 3 where the pruned dictionaries for the four different cut off values of 10%, 15%, 30% and 60% appear in clockwise order starting from the upper left corner. An additional advantage of pruning dictionaries is the potential speedup of the generation of document profiles.

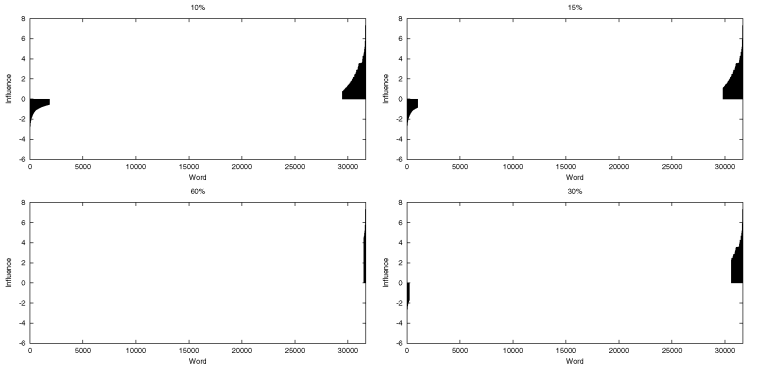

Fig. 3. Different cut off values applied to the dictionary of Figure 2; clockwise from the upper left corner the values are: 10%, 15%, 30% and 60%.

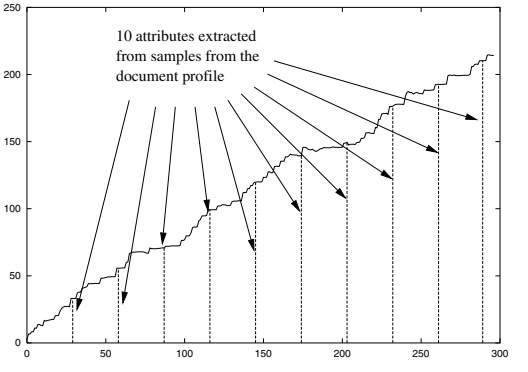

Fig. 4. Reading off attributes from a document profile.

2.3 Turning Document Profiles into Attributes

We have used two different methods to turn document profiles into a constant number of propositional attributes. We need to deal with the fact that the number of words in each document is different, therefore also the length of document profiles differs. The first methods solves that problem by simply reading off the value of the document profile after a certain percentage of the document has been read. Looking at Figure 4, we see that ten values are extracted, with equal-sized gaps in between. In a naive approach the maximum number of attributes that can be extracted in this manner is limited by the size of the smallest document. The second method for extracting attributes is even simpler, computing just some very high-level summary information about a document profile. Specifically, such a description comprises a mere seven attributes: the maximum and the minimum value encountered, the respective positions of these two extrema relative to the document length, a boolean indicator whether the maximum is

Table 2. The ten largest categories from the ModAPTE split.

category	# of training articles	# of test articles
earnings (earn)	2877	1087
corporate acquisitions (acq)	1650	719
money market (money-fx)	538	179
grain (grain)	433	149
crude oil (crude)	389	189
trade issues (trade)	369	117
interest (interest)	347	131
wheat (wheat)	212	71
shipping (ship)	197	89
corn (corn)	181	56

reached before the minimum, and the total number of words for and against the category (i.e. how many words carried a positive influence, how many carried a negative influence). Obviously this is just one of a few possible high-level summary descriptions, other potentially interesting attributes include document length or final value in the profile.

3 Experimental Setup

To investigate the performance of the two-step process described above we have conducted an empirical evaluation using the Reuters corpus[1]. We used the same train-test split as proposed in [13] where a total of 12,902 documents is split into a train-set of 9,603 documents and a test-set of 3,299 documents. We restricted our evaluation to the 10 most common categories, as presented in Table 2. The only pre-processing operation we did was to lower case the characters. We did not use any stemming nor stop word removal techniques.

For our evaluation we used the standard information-retrieval performance measures of precision and recall, as well as aggregate measures based on those two. The aggregate measures were F-measure and the precision-recall mean. The standard F-measure computes the harmonic mean of precision and recall and the precision-recall mean is the arithmetic mean (average) of those two measures. The four formulae are summarised in Table 3. Macroaveraging simply computes averages of either the F-measures or precision-recall means over several categories.

4 Experimental Results

We have conducted an extensive series of experiments to judge the performance of various standard classifiers using document profiles, and also to investigate

[1] The Reuters-21578 collection may be freely downloaded for experimentation purposes from *www.research.att.com/~lewis/reuters21578.html*

Table 3. Four information-retrieval evaluation measures: precision, recall, F-measure and precision-recall mean.

measure name	formula
precision	$\frac{tp}{tp+fp}$
recall	$\frac{tp}{tp+fn}$
F-measure	$\frac{2tp}{2tp+fp+fn}$
precision-recall mean	$\frac{tp(2tp+fn+fp)}{2(tp+fp)(tp+fn)}$

the effects of the dictionary tuning we have described above. For lack of space we will only concentrate some of the findings here, a complete report can be found in the forth-coming Master's thesis of the first author. We used the following classifiers from the Weka [14] package: J48 (C4.5, a decision tree algorithm [15]), OneR (rule based algorithm [16]), IBk (k-Nearest Neighbour (k-NN) [17]), SMO (Support Vector Machine [18]) and Naive Bayes (Naive Bayes algorithm [19]). We have also added a very simple classifier called *Polarity* that simply predicts the sign of the last value in the document profile. Polarity is closely related to *multinomial* Naive Bayes ([20]).

4.1 Which Classifier to Use?

Figure 5 shows the performance of the six different classifiers on the category *trade* per number of attributes taken from the profile. While J48, OneR, IBk and SMO show equivalent results—SMO shows an interesting behaviour, with recall rising at the expense of precision as the number of attributes increases past 100—*Naive Bayes* and Polarity do not seem to be as influenced as the afore-mentioned classifiers schemes by the number of attributes generated from the profile. They show impressive recall, but unfortunately also poor precision. Qualitatively speaking, graphs for other categories look similar.

Figure 6 depicts macroaveraged F-measures of the 6 classifiers on the 10 categories (described in Table 2) per number of samples. Two distinct clusters are noticeable: above 0.65 points F-measure with J48, IBk and OneR, and below with *Naive Bayes*, SMO and Polarity. Overall, J48 and IBk are clearly dominant, followed closely by OneR. The poor performance of SMO, *Naive Bayes* as well as Polarity is probably caused by the high correlation between the generated attributes. Summing up, J48 should be preferred to IBk for the slightly better results and the computationally expensive classification process of the lazy learning approach.

4.2 Pruning the Dictionary Plus Reading only Parts of a Document

In this section we only employ J48, because it performed well in the experiments reported in the last section, and it is fast. Figure 7 illustrates the effect of using a pruned dictionary (on the left hand side) and of only reading initial portions of documents (on the right hand side) using 150 attributes extracted from the

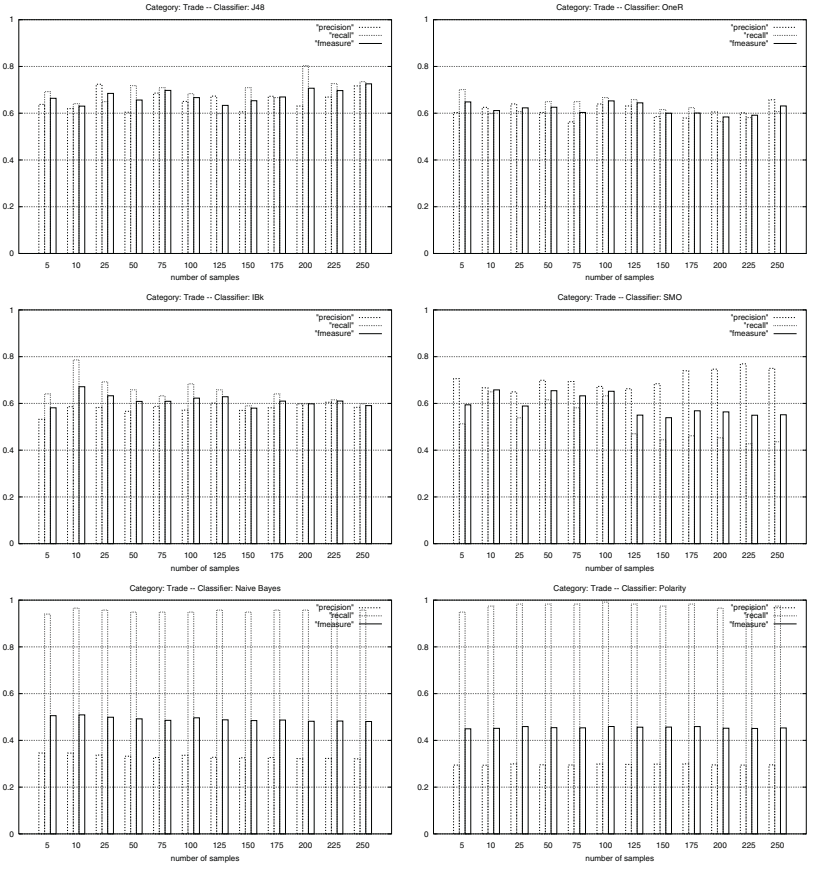

Fig. 5. Performances of six different classifiers on the Reuters category *Trade*.

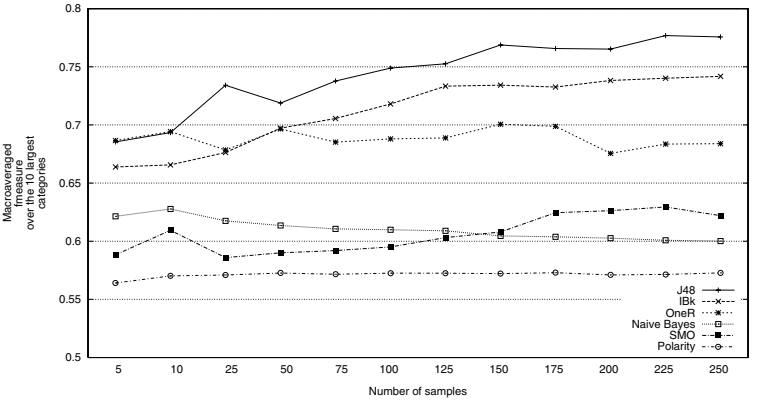

Fig. 6. Macroaverages for 6 classifiers plotted per number of attributes.

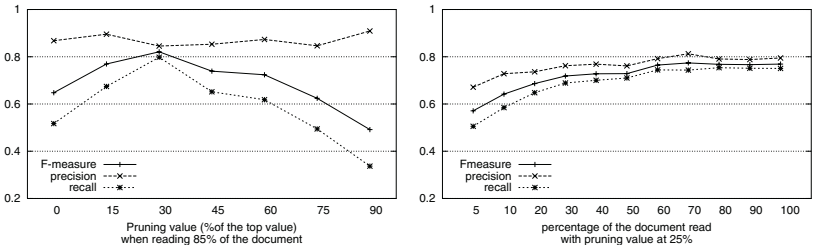

Fig. 7. Effects of reading only initial parts of documents and pruning dictionaries.

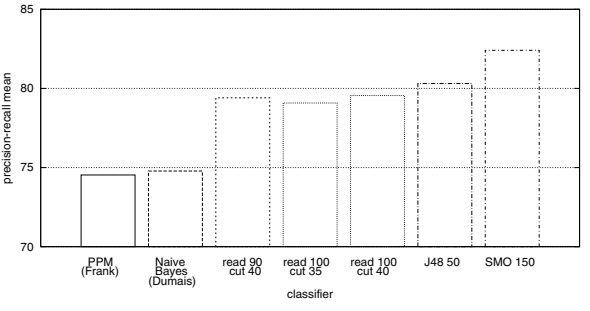

Fig. 8. Comparing 7 classifiers' precision-recall mean macroaverage over the 10 largest Reuters categories.

profile. While the precision is generally not affected by reducing the number of words in the dictionary, recall significantly decreases as the pruning percentage increases. Also note that performance is much less affected by the percentage of the document read than by the size of the pruned dictionary. Furthermore, precision appears to react more robustly than recall to the effect of reducing the size of the portion of the document that is being actually read.

4.3 Performance of the Summary Attributes

In this subsection we compare the performance achievable with the seven-attribute summary information to various standard text classifiers. Figure 8 shows the macroaverage of the precision-recall means of 3 variations of this classifier against results obtained by [3] for *Naive Bayes* and [9] for PPM and against SMO and J48 on the ten largest Reuters-21578 categories. The three variations all used J48 on a shifted dictionary, using either 35% or 40% as a cutoff value, and read either 90% or the whole document. Both SMO and J48 were run on top of a standard bag-of-words-based feature selection using info-gain. 50 features for J48 and 150 features for SMO yielded the best results. The results show that J48 using this tiny set of features outperforms *Naive Bayes* and PPM, closely approaches standard J48, but does not perform as well as SMO.

4.4 Complexity of the Attribute Generation

The algorithm's complexity is equivalent to the complexity of *Naive Bayes* in the sense that it is linear in the number of words and in the number of categories. The complexity of SMO, for comparison, is on average $n.log(n)$. Accessing the dictionary to retrieve influence values is generally $log(n)$, but if necessary, perfect hash functions could be used to reduce this dictionary access cost to a constant. Computing such hash functions will be easier for smaller dictionaries.

5 Conclusion

This paper has presented a text classification approach based on document profiles. Its predictive performance is comparable to more standard approaches, but the method is extremely simple, therefore fast and highly scalable. Our two-step approach effectively transforms a sparse learning problem into a dense one without having to explicitly select single features from the original representation.

The most promising direction for future work will be investigating combinations of the different sets of attributes available. Two approaches are possible: one can combine the high-level summary, the partial sums, as well as standard feature subsets into one larger single set of features. Secondly, single classifiers can be trained on these different feature sets in isolation and then be put together into ensembles. Another direction will be comparing our influence formula with the usual TFIDF document representation. A good starting point will be the thorough study carried out by Rennie et al. [21].

References

1. Yang, Y., Slattery, S., Ghani, R.: A study of approaches to hypertext categorization. Journal of Intelligent Info. Systems **18** (2002) 219–241
2. Roth, D.: Learning to resolve natural language ambiguities: a unified approach. In: Proc. of AAAI-98, 15th Conf. of the American Association for Artificial Intelligence, AAAI Press (1998) 806–813
3. Dumais, S., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization. In: Proc. of CIKM-98, 7th ACM Int. Conf. on Info. and Knowledge Management, ACM Press (1998) 148–155
4. Lewis, D.D.: Naive (Bayes) at forty: The independence assumption in information retrieval. In: Proc. of ECML-98, 10th European Conf. on Machine Learning, Springer (1998) 4–15
5. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Proc. of ECML-98, 10th European Conf. on Machine Learning, Springer (1998) 137–142
6. Rocchio, J.J.: Relevance feedback in information retrieval. The SMART Retrieval System: Experiments in automatic document processing (1971) 313–323
7. Yang, Y., Chute, C.G.: A linear least squares fit mapping method for information retrieval from natural language texts. In: 14th Int. Conf. on Computational Linguistics (COLING). (1992) 447–453

8. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Proc. of ICML-97, 14th Int. Conf. on Machine Learning, Morgan Kaufmann (1997) 412–420
9. Frank, E., Chui, C., Witten, I.H.: Text categorization using compression models. In: Proc. of DCC-00, IEEE Data Compression Conf., IEEE Computer Society Press (2000) 200–209
10. Wiener, E.D., Pedersen, J.O., Weigend, A.S.: A neural network approach to topic spotting. In: Proc. of SDAIR-95, 4th Annual Symposium on Document Analysis and Info. Retrieval. (1995) 317–332
11. Scott, S., Matwin, S.: Text classification using WordNet hypernyms. In: Use of WordNet in Natural Language Processing Systems: Proceedings of the Conf. Association for Computational Linguistics (1998) 38–44
12. Lee, M.D.: Fast text classification using sequential sampling processes. In: Proc. of the 14th Australian Joint Conf. on Artificial Intelligence, Springer (2002) 309–320
13. Apté, C., Damerau, F., Weiss, S.M.: Automated learning of decision rules for text categorization. Information Systems **12** (1994) 233–251
14. Witten, I.H., Frank, E., Trigg, L., Hall, M., Holmes, G., Cunningham, S.J.: Weka: Practical machine learning tools and techniques with java implementations. In: Proc ICONIP/ANZIIS/ANNES'99 Int. Workshop: Emerging Knowledge Engineering and Connectionist-Based Info. Systems. (1999) 192–196
15. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
16. Holte, R.: Very simple classification rules perform well on most commonly used datasets. In: Machine Learning. Volume 11. (1993) 63–91
17. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. Machine Learning **6** (1991) 37–66
18. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: B. SchOlkopf, C. Burges, and A. Smola Advances in Kernel Methods - Support Vector Learning. (1998)
19. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proc. of the Eleventh Conf. on Uncertainty in Artificial Intelligence, Morgan Kaufmann (1995) 338–345
20. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification. In: AAAI-98 Workshop on Learning for Text Categorization. (1998)
21. Rennie, J., Shih, L., Teevan, J., Karger, D.: Tackling the poor assumptions of naive bayes text classifiers. In: Proc. of the 20th Int. Conf. on Machine Learning, Morgan Kaufmann (2003)