In: Knorz, Gerhard; Kuhlen, Rainer (Hg.): Informationskompetenz – Basiskompetenz in der Informationsgesellschaft. Proceedings des 7. Internationalen Symposiums für Informationswissenschaft (ISI 2000), Darmstadt, 8. – 10. November 2000. Konstanz: UVK Verlagsgesellschaft mbH, 2000. S. 145 – 162

Automatic Document Classification:

A thorough Evaluation of various Methods

C. Goller[1], J. Löning[2], T. Will[1], W. Wolff[2]

SAIL-LABS[1]
Balanstr. 57
81541 München
Germany
http://www.sail-labs.de

iXEC[2]
Lilienthalstr. 25
D-85399 Hallbergmoos
Germany
http://www.ixec.com

Email: christoph.goller@sail-labs.de

Abstract

(Automatic) document classification is generally defined as content-based assignment of one or more predefined categories to documents. Usually, machine learning, statistical pattern recognition, or neural network approaches are used to construct classifiers automatically. In this paper we thoroughly evaluate a wide variety of these methods on a document classification task for German text. We evaluate different feature construction and selection methods and various classifiers. Our main results are: (1) feature selection is necessary not only to reduce learning and classification time, but also to avoid overfitting (even for Support Vector Machines); (2) surprisingly, our morphological analysis does not improve classification quality compared to a letter 5-gram approach; (3) Support Vector Machines are significantly better than all other classification methods.

1. Introduction

Document classification is known under a number of synonyms such as *document/text categorization/routing* and *topic identification.* For more work on automatic document classification see e.g. the TREC-conference series [1].

Basically document classification can be defined as *content-based assignment* of one or more *predefined categories (topics)* to documents. Document classification can be used for document filtering and routing to topic-specific processing mechanisms such as information extraction and machine translation. However, it is equally useful for filtering and routing documents directly to humans.

Applications are e.g. filtering of news articles for knowledge workers, routing of customer email in a customer service department, or detection and identification of criminal activities for police, military, or crete service environments.

Manual document classification is known to be an expensive and timeconsuming task. *Machine learning* approaches to classification suggest the automatic construction of classifiers using induction *over pre-classified sample documents.* In this paper we thoroughly evaluate and compare various methods for this kind of automatic document classification.

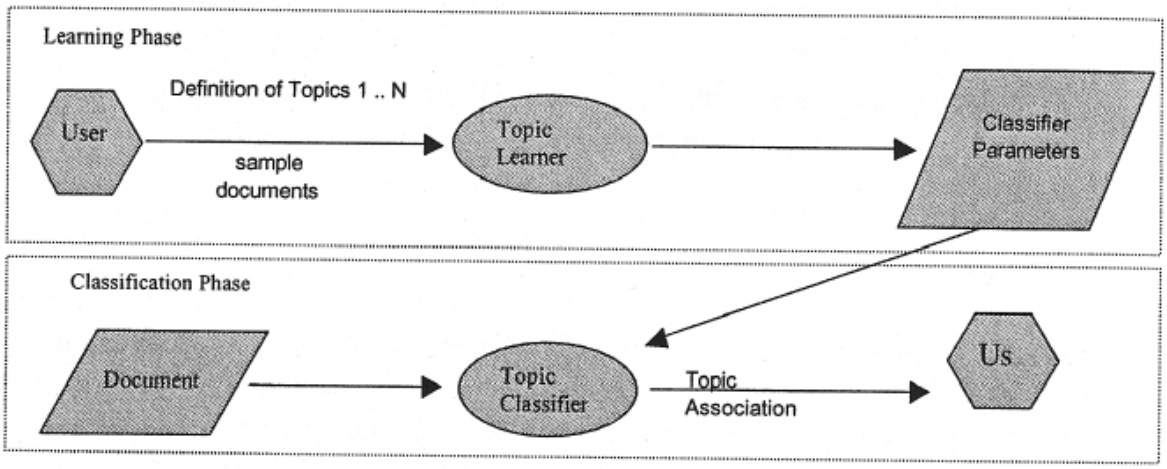


Figure 2: Document Classification

Section 2 defines automatic document classification and relates it to document retrieval. In Section 3 we describe the document corpus used for our experiments. In Section 4 we introduce the different approaches that we have selected and evaluate them from a theoretical point of view. The experimental comparison follows in Section 5. We summarize our results in Section 6 and present some ideas for further improving automatic document classification.

2. Automatic Document Classification

We can distinguish two phases in automatic document classification, the *learning phase* and the subsequent *classification phase* (see Figure 1). In the *learning phase* users define categories (topics) in which they are interested (their *information need)* by giving *sample documents (training examples)* for each of these categories. Most methods for automatic document classification also require *counterexamples* for each category that is sample documents that do not deal with the respective topic.

In a standard application for automatic document classification like news filtering users assign categories to the documents of a selected collection by hand. Documents may be assigned to more than one category if they deal with several of the topics or if one has a hierarchy of topics. The document collection used for learning should be as representative as possible for the documents that one

expects to encounter in the future. All documents that are not assigned to a category serve as counterexamples for this category.

The topic learner component analyzes the sample documents and tries to identify the content that is specific for each category. This analysis is essentially an *inductive reasoning* process which involves *generalization* and *abstraction.* The output is a *model* for each category which is represented by a set of *classifier parameters.* Most classifiers make *a priori assumptions* about the underlying model and its *complexity. If* the assumed model complexity is too high, *overf tting* can occur. This means that the model overspecializes with respect to the training examples and generalization to new previously unseen examples becomes bad. Overfitting is one of the biggest problems in machine learning. Normally one assumes that the learning phase is invoked very rarely. Therefore efficiency is not *of* primary importance.

In the *classification phase* new (previously unseen) documents can be given to the topic classifier which returns a topic association (a rating or classific a-

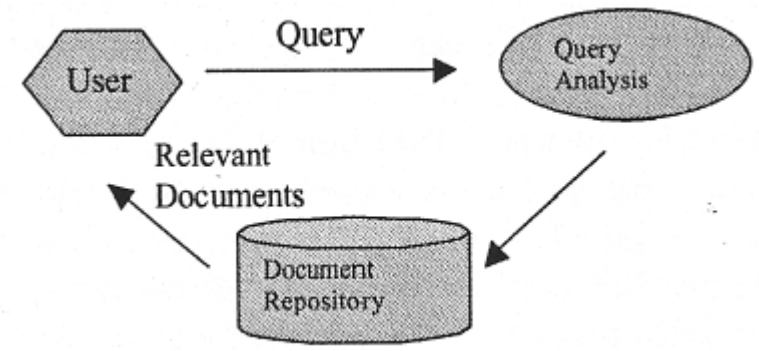


Figure 3: Document Retrieval

tion for each topic). Basically, it tells the user whether the document deals with the training topics and especially with which topics it deals. It is assumed that a lot of documents have to be classified. In a news filtering application new messages constantly arrive and have to be filtered or routed. Therefore efficiency is very important for the classification phase.

Automatic document classification is closely related to *document retrieval* (see Figure 2). In document retrieval the user specifies his *information need* be giving a *query,* which is analyzed and then applied to a relatively fixed and preprocessed (indexed) document corpus. The relevant documents are presented to the user.

There are two main differences between automatic document classification and document retrieval. The First difference is a rather technical one. In document retrieval the documents which are compared to the users information need are typically members of a big and relatively fixed document corpus which is preprocessed (indexed) independently of the individual queries. Normally, queries are small. They can be applied to the whole corpus in parallel and do not have to be applied to every document sequentially. Usually a lot of queries have to be answered and new queries constantly arrive. Therefore query analysis also has to

be very efficient. In document classification, the users' information need (topics) is relatively stable. However, one does not have a fixed document collection which can be preprocessed. Normally documents have to be processed sequentially by the classifier.

The second difference concerns the way how the information need is specified. In document retrieval the users' *information need is* specified by usually small queries (keywords). The users have to know exactly what they are interested in. Differences in language uses between authors of documents and users who specify queries offen lead to low recall. Therefore, query expansion by using thesauri or domain models is very important. In document classification users specify their information need by selecting documents as examples and counterexamples for a topic. If there are enough sample documents and if they are representative for the kind of documents usually encountered, classification will be quite good. The users don't have to know exactly what they are looking for and the use of thesauri or domain models is less important. However, this difference between document classification and retrieval disappears if we consider document retrieval with user feedback. Here the users are allowed to refine their original queries by selecting relevant an non-relevant documents from the documents that are presented by the retrieval system as answer to their original query.

3. Test Corpus

As test corpus for automatic document classification we used a subset of articles from the German newspaper "Süddeutsche Zeitung (1998)" (SZ). The CD-Rom version [2] of the SZ contains topic-labels for most of the articles. Some articles are labeled with more than one topic. From the articles from the "München" section (local news) we selected those 20 topics with the highest numbers of articles (see Table 1). The total number of articles used in the experiments is 1083. This is less than the sum of column 2 from Table 1 due to multiple topic assignments for some of the articles.

Topic	#of articles	Topic	#of articles
Raubüberfälle	112	Flughafen München	47
Abschiebung von "Mehmet"	95	Mordversuche	45
Das ganze Leben ist ein Test	79	Verkehrsunfälle	45
Diebstähle	71	Betrugsdelikte	43
Oktoberfest	62	Adventskalender der SZ	42
Mordfälle	62	S-Bahn in München	39
Umbau Olympiastadion	60	Ausländer	37
Uhl, Hans-Peter	52	Illegaler Drogenhandel	37
Brände	52	Landtagswahl 1998	36
SZ-Serie Vier für München	51	Kiest, Erich	35

Table 1: Test Corpus

4. Selected Approaches and Theoretical Evaluation 4.1 Feature Construction

In Section 2 we define automatic document classification as an inductive reasoning task. This of course suggests the use of classification techniques from the fields of machine learning, statistical pattern recognition and neural networks. However, almost all existing learning and classification techniques require vectors of (real) numbers as input. They cannot work directly an documents (text). Therefore, vector representations of documents have to be constructed in order to make these methods applicable. The process of constructing these vector representations is called *feature (vector) construction.* Feature construction methods generally differ in the amount of linguistic and statistic sophistication that is applied. We compared two kinds of features, viz. letter 5-gram features and features constructed by a morphological analysis.

4.1.1 Letter 5-gram Features

For our ferst approach we generated letter 5-grams without applying any tokenization. Special characters like "!%&$" and whitespaces were included. We did not distinguish upper and lower case characters. The reason for including inter-word 5-grams is that in this way at least some multiword phrases (e.g. 'New York' in the example sentence below) are represented. The reason for including whitespaces is that we wanted short important strings like " IBM " (that are always included in inter-word 5-grams) not to become unrecognizable because of their varying context. For the sentence ' New York ist groß.', we get the following 5-grams: 'new y' 'ew yo', 'w yor', 'yourk', 'york ' ork_i', 'rk_is', kist', 'groß', 'groß.

Topic	1	2	3	4	5
Abschiebung von "Mehmet"	ehmet	mehme	hmet"	met"_	"mehm
Adventskalender der SZ	ür_gu	skate	ute_w	erke	adven
Ausländer	ölker	völke	bevö	evölk	lkeru
Betrugsdelikte	etrug	betru	trugs	ädigt	aatsa
Brände	brand	bran	euerw	rwehr	erweh
Diebstähle	einbr	einb	dieb	stohl	estoh
Flughafen München	ghafe	hafen	flugh	lugha	ughaf
Illegaler Drogenhandel	rogen	ealer	droge	drog	deale
Kiesl, Erich	lesl_	h_kie	ch ki	kiest	kies
Landtagswahl 1998	agswa	wahl	wähl	ähler	tagsw
Mordfälle	mord	mord	s_opf	pfers	es_op
Mordversuche	mord	hlags	bensg	tots	totsc
Oktoberfest	wiesn	wies	berfe	iesn-	oberf
Raubüberfälle	berfa	erfal	überf	rfall	räube
S-Bahn in München	bahn-	s-bah	-bahn	_s-ba	e_s-b
Das ganze Leben ist ein Test	_test	r_tes	prüf	klass	bewer
SZ-Serie Vier für München	ler_f	ür mü	für m	vier	r mün
Uhl, Hans-Peter	_uhl	r_uhl	er uh	_uhl	hans-
Umbau des Olympiastadions	stadi	adion	tadio	_umba	umbau
Verkehrsunfälle in München	prall	unfal	Hetz	erlet	unfa

Table 2: The five most relevant 5-gram features for each topic

4.1.2 Morpheme Features

For our second approach we implemented a simple *morphological analyzer* for German which combines *inflectional* and *derivational stemming* and a *word compound analysis.* After *tokenization* tokens are analyzed as concatenations of the following four morphological categories:

	categoty	examples		lexicon entries[1]
		Verurteilung	aber	
v	prefix	ver		111
s	(word-)stem	urteil		6.614
e	suffix	ung		97
t	solitary stein		aber	132

An admissible analysis is subject to the following two restrictions:

words analyzed as t consist of just this morpheme (no pre- or suffixes): p = t otherwise, the analysis of words follows the regular pattern p:

$p=w^{+}$ and$w^{=}v^{*}s^{+}e^{*}$

In order to tune the morphological analyzer heuristics concerning the ratio between the morphological categories in an analyzed token were developed.

We tested our morphological analyzer on the Munich part of the SZ (176000 unique tokens). For our test 1000 tokens were randomly selected. The analyzer was able to analyze 77.4% of the test tokens. 91.3% of the analyzed words were classified as correctly analyzed by humans. 5.1% of the analyzed tokens were proper names or misspelled words that were analyzed nonetheless. For 3.6% the analysis was incorrect. The 22.6% of the tokens that could not be analyzed by our morphological analyzer were mainly irreducible words such as names, abbreviations, monosyllables, foreign words, misspellings, dialect words, or numbers. Only 7.7% of the tokens that could not be analyzed could be analyzed by humans.

[1] The starting point for prefix, suffix, and stein lexicons were lexicons developed for hyphenation by the Technical University of Vienna [3]. However, we considerably extended and modified these lexicons.

Topic	1	2	3	4	5
Abschiebeng von	mehmet	mehmets	eltern	ausweis	türk
Adventskalender der SZ	kalend	adventkalen	advent	Süd-deutsche	hilfwerk
Ausländer	völk	bevölk	Länder	ausländer	multikultu
Betrussdelikt	betrug	tru	eeschäd	reu	zweihalb
Brände	Brand	feuer	feuerwehr	lösch	wehr
Diebstähle	dieb	stohl	eestohl	einbrech	stahl
Fluthafen München	hafen	flughafen	flus	airport	fine
Illegaler Drogenhandel	deal	drog	rauschgift	rausch	heroin
Kiesl, Rich	kiesl	erich	altober	kiesls	altob
Lantagswahl 1998	tagswahl	wähl	wahl	stimmkreis	landtag
Mordfälle	mord	töt	Mord-kommis	kommüss	opfer
Mordversuche	mord	Mord-versuch	mordsuch	stoch	totschlag
Oktoberfest	wiesn	Oktoberfest	zelt	oktober	bier
Raubüberfälle	überfall	raub	räuber	beut	tät
S Bahn in München	fahrgäste	verspät	stamm-streck	streck	bahnhof
Das ganze Leben ist ein	test	prüf	bewerb	schul	klass
SZ-Serie Vier für München	vier für	uns	ia	was	häus
Uhl, Hans-Peter	hanspete	uhl	peter	hans	csu
Umbau des Olympiastadion	stadion	o.-stadion	umbau	olympia	architekt
Verkehrsunfälle	unfall	schleuder	prall	erfaßt	verletz

Table 3: The five most relevant morpheme features for each topic.

From every analyzed token we generated the following features: (1) the whole token without suffix; (2) every stein with its associated prefix and without suffix; (3) every stein; (4) 2-grams of stems. Furthermore, every irreducible token became a feature by itself. In order to represent *multiword phrases,* we additionally included those inter-word 2-grams of stems which had a very high frequency in our corpus.

4.1.3 From Features to Feature Vectors

The features described in Sections 4.1.1 and 4.1.2 are strings of characters. The number of unique features in the document collection determines the *dimension* of the feature vector representations for the documents and the Position of each feature in an alphabetically ordered list of all features determines its position in the feature vector representations. *A feature vector representation* for a document is

simply a vector of weights for all the features[2].

We started our experiments with *binary weights* (only using the information whether a feature occurs in a document or not). However, in a first evaluation phase we found that using the *term frequency (TF),* which is the number of occurrences of the feature in the document, improves classification quality for all classifiers. We also found that using the product of term frequency and *inverse document frequency* (IDF = # documents in the collection / # documents that contain the feature), which is a very popular weighting scheure in document retrieval, improves classification quality for the *Centroid classifier* and the *Support Vector Machine* considerably, while it does not change results for the other classifiers[3]. For Support Vector Machines a refinement (the square root of TFIDF) further improves classification quality. For a brief description of the classifiers that we used see Section 4.3. For a discussion of TFIDF see e.g. [4].

4.1.4 General Comparison of N-gram Features and Morpheme Features

N-gram letter features have a variety of advantages compared to morpheme features. Implementation of n-gram feature construction is very easy and *independent of language.* N-gram features automatically perform certain kinds of *stemming* and they are robust against *misspellings.* Furthermore, N-gram features automatically capture many kinds *of multiword phrases,* if one considers n-grams across word-borders (inter-word n-grams).

Morphological analysis is of course *language-specific* and it is quite an effort to implement a morphological analyzer. One has to develop the lexicons, take care of incorrectly analyzed words and ambiguities, and one has to avoid both, *over-* and *under-stemming.* For German, *word compound analysis is* also very important. Furthermore, a morphological analyzer consumes more computational resources than n-gram analysis.

However, there are also big advantages which speak for morpheme features. It seems clear that morpheme features contain more information than n-gram features, since they represent *meaningful word-constituents* even if these constituents are longer then n characters. Therefore, classifiers built from morpheme features are more *understandable* than those built from n-gram features. Normally, there are considerably fewer unique morpheme features than e.g. 5-gram features for a given document and morpheme features are less correlated than n-gram features. The *lower dimension* of morpheme feature vectors and the *lower correlatedness* make feature selection / dimensionality reduction and learning easier. In our experiments the *inverse file index* of the test collection for 5-gram features is five times bigger than for morpheme features leading to an increase in feature selection costs of a factor five. Tables 2 and 3 list the five most important features for each of our test topics determined by the mutual information measure (see Section 4.2). The higher correlatedness of 5-gram features can easily be

[2] If a new document (classification phase) contains features not encountered in the document collection, these features are simply ignored.

[3] Interestingly, even the Perceptron was not able to profit from the additional IDF-weighting.

recognized since for several topics the five most relevant 5-gram features originate from only one word. For an experimental comparison of 5-gram and morpheme features see Section 5.4.

4.2 Feature Selection / Dimensionality Reduction

For our test corpus about 30000 unique morpheme features and about 185000 unique 5-gram features were generated. Most classification methods cannot handle such *high-dimensional input* (computational costs for learning and/or classification become intractable). A further problem is that the model complexity for many classifiers increases with the dimension of their input. This means that high-dimensional input vectors can cause overfitting. Therefore, *dimensionality reduction* or *feature subset selection* methods have to be applied. There are a lot of methods for dimensionality reduction around in the statistics literature. One of them is *principal component analysis,* in which orthogonal (empirically independent) linear combinations of the original features are determined which have the highest variance. In the document retrieval and classification area this approach is known as *latent semantic indexing [5].* However, the computational costs are very high.

We decided to use a feature subset selection method. This means that for each topic a subset of the most relevant features is determined. In order to determine the relevancy of a feature with respect to a topic, we used the *mutual information* measure. The definition of mutual information can be found in every textbook on information theory. Basically, the mutual information between a feature and a topic tells us how much information the feature provides for the topic (and vice versa). Of course one does not know the mutual information between features and topics. We estimated the mutual information between the property that a document belongs to the considered category and the property that the document contains the considered feature. Probabilities were estimated using empirical relative frequencies. Tables 2 and 3 list the five most relevant features found for our test topics in this way.

Furthermore, we considered two extensions of feature subset selection with mutual information: *FuCE* (Feature subset selection using Classifier Errors) and *Autofeatures.*

FuCE works as follows. First a set of relevant features is constructed for every topic in the way described above. The classifier is trained using these features. Those documents of the training set which the classifier does not classify correctly are used to extend the set of features. *False positives* and *false negatives* are used separately. For *false positives* we determine features that distinguish them from real positives. In the same way we determine features that distinguish *false negatives* from real negatives. For both tasks we again used mutual information.

Feature subset selection requires to determine the number of features one wants to use and we extensively explored how the classification performance depends on the number of selected features (Section 5.3). With *Autofeatures* we

implemented a method that determines the number of relevant features for feature subset selection in *a topic-dependent way.* Features are selected until the sum of their individual mutual information with the topic is greater than the entropy of the topic. This takes into account that if there are some very unique features for a topic, only these few features are selected. On the other hand, if there are no features containing much information for the topic, this method selects a bigger subset of. The problem with this method is, that features normally are not independent of each other. Therefore, their joined mutual information with a topic is normally much smaller than the sum of their individual mutual information with the topic. However, the joint mutual information cannot be computed since the statistical data are normally not sufficient. In our experiments we therefore added features until the summed mutual information exceeded the entropy of the topic times a factor (2, 4, 6, 10, 20, 30, 50).

4.3 Classification Methods

4.3.1 Selected Classification Methods

We compared the following classification methods in our experiments:

- Perceptrons (standard Perceptron algorithm that produces a linear discrimination function),
- discrete Naive-Bayes (P(clx)—P(xjc)P(c) = flP(xi(c)P(c), if xi independent),
- MC4 (decision tree similar to the well-known ID3),
- 3 Nearest-Neighbor (the new vector is assigned to the dass of the majority of its nearest neighbors),
- Rocchio Centroid (very simple linear classifier based on the difference between the means of positive and negative examples), and
- Support Vector Machines.

Perceptron, Naive-Bayes, MC4, and Nearest-Neighbor were taken from the MLC++ environment [6] and used with their default parameter settings. The Rocchio Centroid is well-known in document retrieval [7]. For the support vector machine we used the implementation from [8] with linear kernels.

4.3.2 General Comparison of Classification Methods

Perceptrons, Rocchio Centroids, and Support Vector Machines with linear kernels produce linear classifiers (hyperplanes separate the classes in feature vector space). Of course this is a strong restriction in *expressivenes* of the classifiers (a strong *model assumption).* However, experience has shown, that most document classification problems are linearly separable. As far as *learning* is concerned, the Rocchio Centroid does not have much theoretical foundation. *The Perceptron*

learning rule seems much more appropriate since it minimizes the number of incorrect classifications. Support Vector Machine learning has the best theoretical foundation. Support Vector Machines are less susceptible to overfitting than other learning methods since the model complexity is independent of the feature space dimension. The model assumptions of Nearest-Neighbor, Naive-Bayes (independence assumption), and decision trees (local decisions) can also be critized. However, they are different from the assumption of linear separability and therefore a comparison is interesting.

The Nearest-Neighbor does not do any learning at all. It simply stores the training examples. *Learning costs* for the Rocchio Centroid are very low, even with very high dimensions. Perceptron and Support Vector Machine learning is tractable, even for high dimensions, though it is more expensive than Rocchio learning. Naive-Bayes and decision tree learning is tractable only for low dimensions at least in the implementations we used (see also Section 5.2).

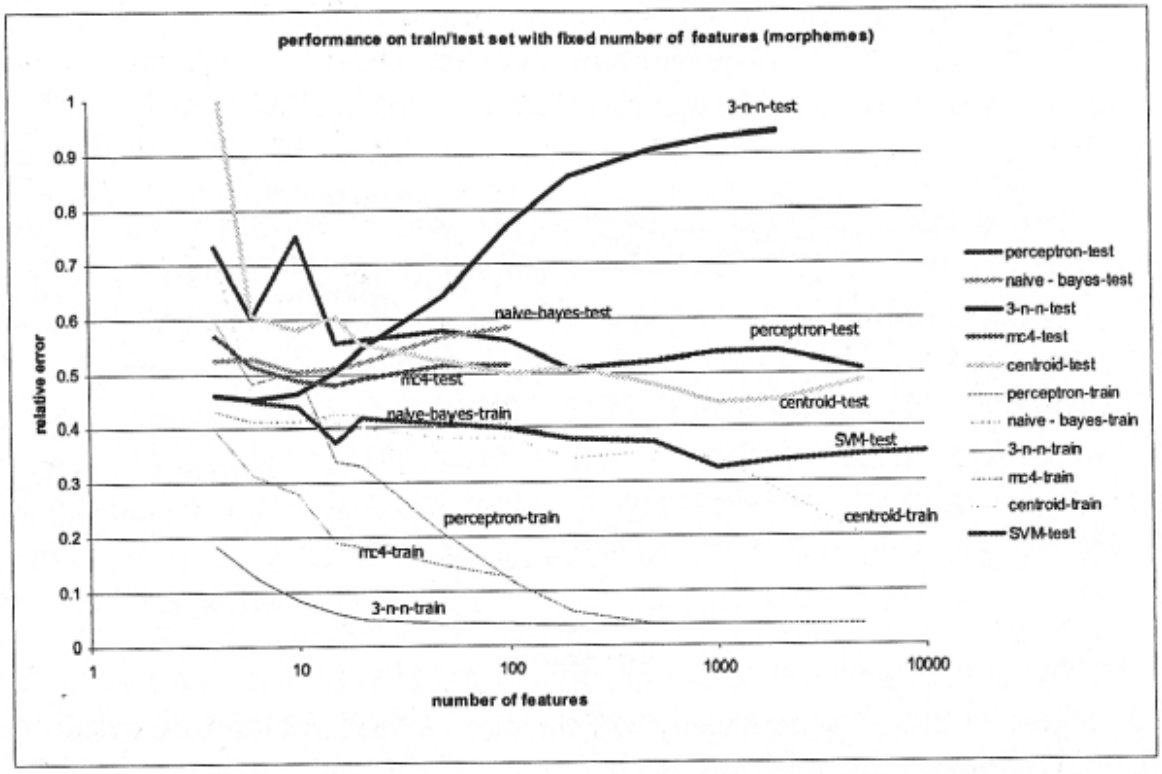
Classification costs are quite low for all classifiers except for the Nearest-Neighbor where the new feature vector has to be compared to all training ex-amples. The classifiers are all reasonably small except for the Nearest-Neighbor classifier, which grows with the number of training examples.

The decision tree has the best *explanation capabilities,* however, inspecting the other classifiers can also be helpful in order to understand how the Classification is done.
A lot of previous studies have shown that k-Nearest-Neighbor and Rocchio Centroid are both very suitable for document Classification. It can be said that they represent the State of the art in this field. However, recent studies (on Reuters benchmarks) indicate that Support Vector Machines might be supe-rior [9].

5. Experimental Comparison

5.1 General Remarks on the Experimental Setup

In order to test the generalization capabilities of the different Classification methods, the test corpus (Section 3) was split into two disjoint sets of docu-ments (a *training set* and a *test sei)* which both contained approximately 550 documents. The split was stratified in the sense that the original ratio of the topics was preserved. Only the training set was used for feature subset selection and learning.



Figurc 3: Classification Quality on Training and Test Set with Standard Feature Subset Selection for Morpheme Features

As error measure we computed the *relative error",* that is the number of falsely classified texts divided by the number of texts in the given topic (positive examples), averaged over all 20 topics (macro averaging). Since the number of positive and negative examples differs about one order of magnitude, this seems to be a more meaningful measure than accuracy without the problems of the *F-measure* [10].

In cases where very expensive experiments (like training MC4 with more than 1000 features) aren't expected to show interesting results, we omitted these experiments.

5.2 Classification Methods

Figure 3 shows the Classification quality on training and test sets for morpheme features with the Standard feature subset selection method. We use these results to compare the different Classification methods. The Autofeature selection method and n-gram features lead to the same results as far as the comparison of Classification methods is concemed.

Support Vector Machines produce the best results both on test and training sets, are robust with respect to the number of features and very fast at training and at Classification. The only drawback seems to be the complexity of implementation.

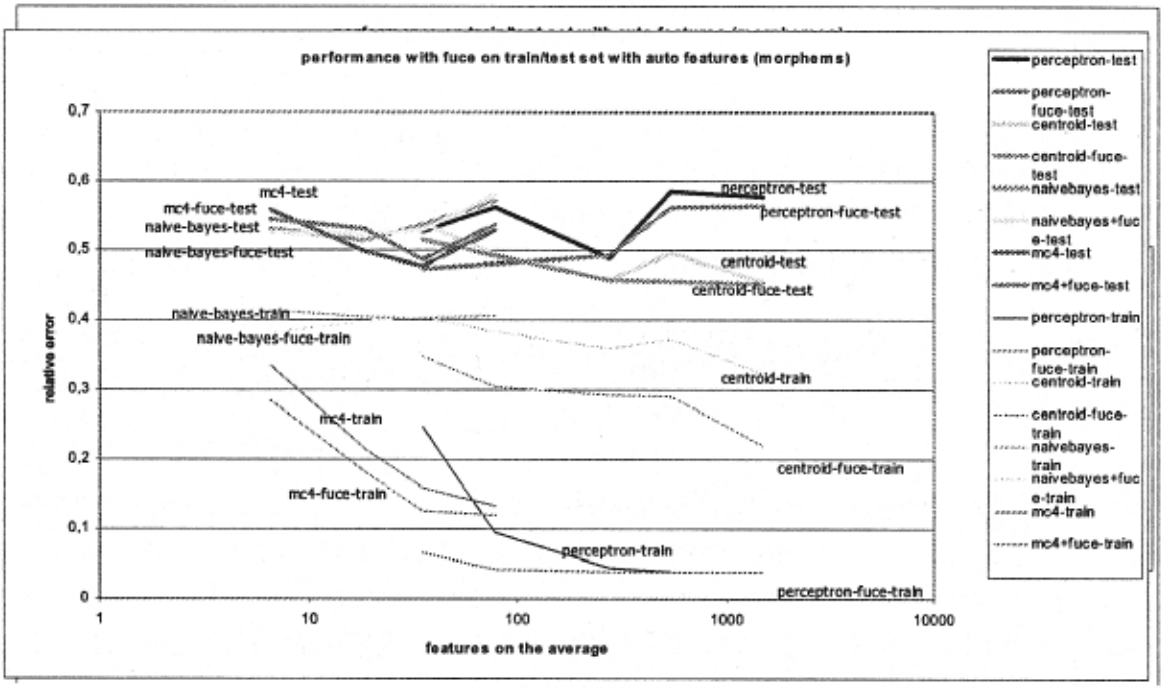


Figure 5: Classification Quality on Training and Test Set with FuCe Feature Subset Selection for Morpheme Features

The Centroid has the second best test set - Performance with over 1000 features, is very simple and fast and is the only algorithm suitable for retrieval, as it works also if fhere are only positive examples for a category. The Performance also seems to be relatively robust to the number of features used. A drawback is it's poor Performance on the training set.

The Perceptron does a good job both on test and training set with approxi-mately 500 features. A disadvantage are the relative high training costs. An optimization by making use of sparseness of feature-vectors may help.

It is interesting to see that with very few features, Nearest-Neighbor and the MC4 do perform nearly as good on the test set as the Centroid with 1000 features and far better on the training set. Note that the Nearest-Neighbor even with few features is relative slow (10 seconds Classification time with 10 features, 550 texts, on a Pentium II - the centroid takes only 1 second with 500 features). MC4 is fast to train with few features and especially fast at Classification time.

5.3 Feature Subset Selection

In Figure 3 we see that Classification quality on the test sets decreases for all classifiers if too many features are used. This happens with very few features for the 3-Nearest-Neighbor, the Naive- Bayes, and MC4. However, it also happens for the Centroid and the Support Vector Machine for more than 1000 features. If we look at the consistently good results on the training sets, the only explanation for the decrease on the test set is overfitting. We think that the effect of overfitting has

not been investigated enough for document Classification so far. Our results indicate that feature subset selection is essential in order to avoid overfitting.

In Figure 4 the results with the Autofeatures subset selection method are shown. The relative errors are plotted against the average number of features that this method computed for factors between 2 and 50. The vector space methods Centroid, Perceptron and Support Vector Machine are not too sensible to changing the numbers of features for high dimensions. Therefore, it does not surprise that the Autofeature method leads to similar results as the Standard subset selection method in high dimensions. On the other hand, with few features we get an improvement for some classifiers (e.g. the Centroid) with the Autofeature method.

Figure 5 shows our results achieved with the FuCe subset selection method. For each classifier we show only results with feature vector dimensions that are suitable for the classifier: higher dimensions for the vector Space methods, lower ones for MC4 and Naive-Bayes. For the Support Vector Machine we omitted these experiments because we didn't integrate the Software fully in our environment. FuCE almost everywhere improved training set Performance. Test set Performance only improved in some cases for the Perceptron and the Centroid.

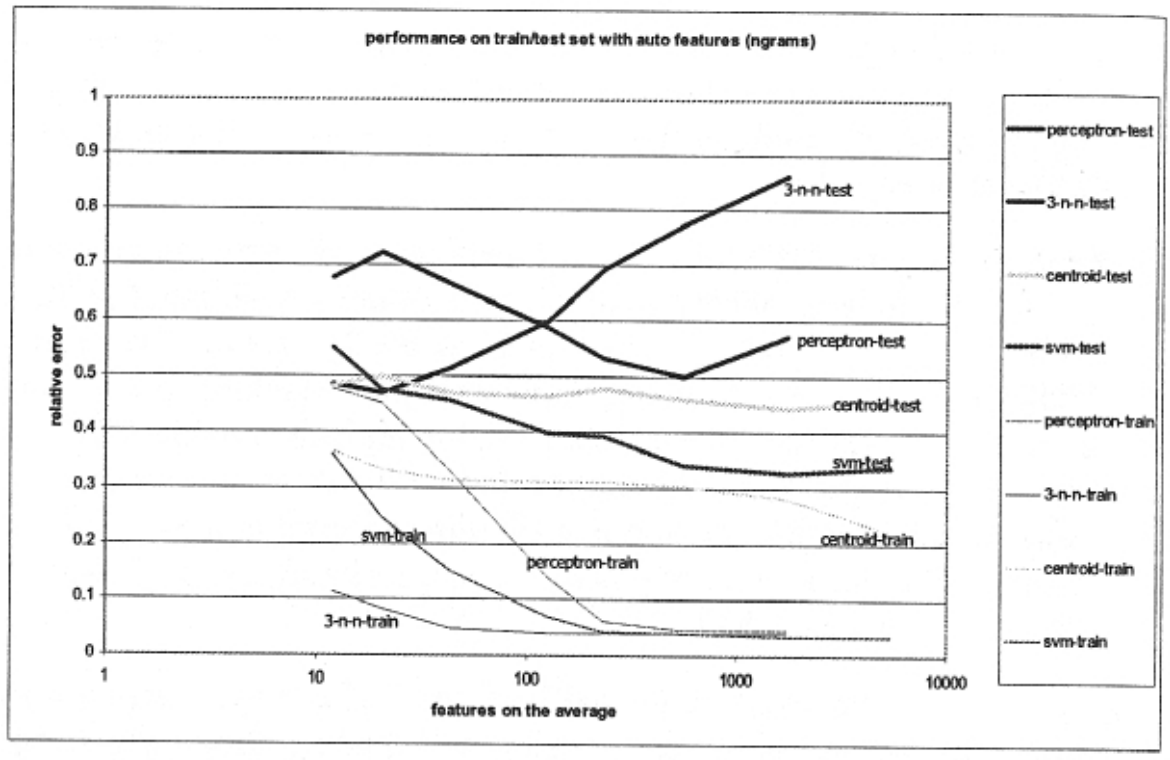


Figure 6: Classification Quality on Training and Test Set with Autofeatures Subset Selection for 5-Gram Features

5.4 N-grams vs. Morphemes

Figure 6 shows our results achieved with 5-gram features. The Classification quality is more or less comparable to the one achieved with morpheme features (Figure 4). However, Perceptron and Support Vector Machine need less

morpheme features than 5-gram features to achieve an identical Classification quality. Our explanation for this is that morpheme features contain more In-formation than 5-gram features. Surprisingly, with few features the Centroid with 5-gram features produces better results than with morpheme features.

5.5 Significance of Results

To get a feeling for the significance of our results, we computed the probabil-ity *P* that one classifier is at least the observed number of times better than the other, under the assumption this number is binomially distributed (with prob-ability of success= 0,5). Note that *P* can be interpreted as the probability that the performance difference between the two methods happens by chance. Here are the results for some cases from Figure 4: For the Centroid and Perceptron, the Centroid had a better test set Performance from 100 to 1500 features, this corresponds to the 4 rightmost data-points in Figure 4. For the three points with the highest difference, *P* was around 0,05. At the third data point from the right, *P* was 0,24. For the Centroid and the Support Vector Machine, we addressed the fourth data-point from the right where these two classifiers show the smallest difference. Here, *P* is below 0,02. Results with a *t-test* are similar. We conclude that for the given corpus, our results are sufficiently significant.

6. Conclusion

In this paper we have presented a thorough evaluation of different approaches for document Classification. We have confirmed recent results about the supe-riority of Support Vector Machines on a new test set. Furthermore, we have shown that feature subset selection or dimensionality reduction is essential for document Classification not only in order to make learning and Classification tractable, but also in order to avoid overfitting. It is important to see that this also holds for Support Vector Machines. Last but not least, we have shown that linguistic preprocessing not necessarily improves Classification quality.

We are convinced that in the long run linguistic preprocessing like a morpho-logical analysis pays off for document Classification as well as for document retrieval. However, this linguistic preprocessing probably has to be more so-phisticated than our simple morphological analysis. A big advantage of linguistic preprocessing compared to n-gram features is that integration of *thesauri, concept nets,* and *domain modeis* becomes possible.

Besides linguistic sophistication, statistics can also help to produce good features. For the future we plan to evaluate different methods for finding topic-relevant *collocations* and *multi-wordphrases.*

Furthermore, we think that feature selection with mutüal information can be improved. There should be a better way for approximating the Joint mutual information of a feature set with a topic, than simply summing up the individ-ual mutual information of every feature. Considering feature pairs probably helps a lot. Furthermore, one should include the *term frequency* into the com-putation of mutual information.

Acknowledgements

At the time when this work was carried out, the authors worked for iXEC (Executive Informatiopn Systems GmbH). Writing this paper took place when two authors had changed to SAIL-LABS (Speech and Artificail Intelligence Labs) where they currently work in the content group.

References

TREC (Text Retrieval Conference): http://trec.nist.gov/

Süddeutsche Zeitung
http:// www.sueddeutsche.de, http://www.diz-muenchen.de/cdrom.htm

TU Wien: Sichere Sinnentsprechende Silbentrennung
http://www.apm.tuwien.ac.at/research/SiSiSi-dt.html

Gerard Salton, and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval.* New York: McGraw-Hill.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. Landauer, and R. Harshman. 1990. *Indexing by Latent Semantic Analysis.* Journal of the American Society for Information Science 41:391-407.

Kohavi and Sommerfield, 1996, *"MLC++ Machine Learning library in* C++", available at http://www.sgi.com/Technology/mlc/

J. Rocchio. 1971. Relevance Feedback in Information Retrieval. In G. Salton, editor, The SMART Retrieval System: Experiments in Automated Document Processing, pages 313-323. Prentice Hall Inc.

T. Joachims. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

T. Joachims. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features.* Proceedings of the European Conference on Machine Learning, Springer, 1998.

D. D. Lewis. 1995. Evaluating and Optimizing Autonomous Text Classifcation Systems, SIGIR Conference 95.