On the Impact of Document Representation on Classifier Performance in e-Mail Categorization

Helmut Berger[(1)], Monika Köhle[(2)], Dieter Merkl[(3)]
[(1)]Electronic Commerce Competence Center EC3, Wien, Austria
[(2)]Institut für Rechnergestützte Automation, Technische Universität Wien, Austria
[(3)]School of Computing and IT, University of Western Sydney, NSW, Australia
helmut.berger@ec3.at, monika.koehle@inso.tuwien.ac.at, d.merkl@uws.edu.au

Abstract: This paper provides an analysis of multi-class e-mail categorization performance. In order to investigate this issue, the quality of various classification algorithms based on two distinct document representation formalisms is compared. In particular, both a standard word-based document representation as well as a character n-gram document representation is used. The latter is regarded as highly noise-tolerant and was originally proposed for automatic language identification and as a convenient means for producing compact document indices. Furthermore the impact of using available e-mail specific meta-information on classification performance is explored and the findings are presented.

1 Introduction

The task of automatically sorting documents of a document collection into categories from a predefined set, is referred to as *text categorization*. Text categorization is applicable in a variety of domains: document genre identification, authorship attribution, survey coding, to name but a few [Seb02]. One particular application is categorizing e-mail messages into legitimate and spam messages, i.e. *spam filtering*. Androutsopoulos et al. compare in [APK$^+$00] a *Naïve Bayes* classifier against an *instance-based* classifier to categorize e-mail messages into spam and legitimate messages, and conclude that these learning-based classifiers clearly outperform simple anti-spam keyword approaches. However, sometimes it is desired to classify e-mail messages in more than two categories. Consider, for example an e-mail routing application, which automatically sorts incoming messages according to their content and routes them to recipients that are responsible for a particular topic. The study presented herein compares the performance of different text classification algorithms in such a multi-class setting. More precisely, the performance of three different text classifiers, when used to categorize e-mails into a manually predefined set of multiple classes, is evaluated. By nature, e-mail messages are short documents containing misspellings, special characters and abbreviations. This entails an additional challenge for text classifiers to cope with "noisy" input data. To classify e-mail in the presence of noise, a method used for language identification is adapted in order to statistically describe e-mail messages. Specifically, character-based n-gram frequency profiles, as proposed in [CT94], are used as features which represent each particular e-mail message. The comparison of the

performance of categorization algorithms using character-based n-gram frequencies as elements of feature vectors with respect to multiple classes is described. In [PS03] a related approach aims at authorship attribution and topic detection. In this paper, the performance of a *Naïve Bayes* classifier combined with n-gram language models is evaluated. The authors mention, that the character-based approach showed better classification results than the word-based approach for topic detection in newsgroups. Their interpretation is that the character-based approach captures regularities that the word-based approach is missing in this particular application.

Besides the content contained in the body of an e-mail message, the e-mail header holds useful data that has impact on the classification task. The study presented in this paper explores the influence of header information on classification performance. Two different representations of each e-mail message were generated: one that contains *all* data of an e-mail message and a second, which only consists of textual data found in the e-mail body. So, the impact on classification results when header information is discarded can be demonstrated.

The remainder of this paper is organized as follows. Section 2 defines the notion of n-grams and describes how frequency profiles of text are generated. The feature selection metric and text categorization algorithms used for this study are reviewed in Section 3. In Section 4 we provide a description of the experimental results for multi-class e-mail categorization. Finally, Section 5 contains a discussion of the experiments.

2 N-gram Frequency Statistics

An n-gram is an n-character slice of a longer character string. When dealing with multiple words in a string, the blank character indicates word boundaries and is usually retained during the construction of the n-grams. However, it might get substituted with another special character. As an example for $n = 2$, the character *bi*-grams of *"topic spotting"* are $\{$*to, op, pi, ic, c␣, ␣s, sp, po, ot, tt, ti, in, ng*$\}$. Note that the "space" character is part of the alphabet in this example and represented by "␣".

Formally, let $\mathcal{A}$ be an alphabet of characters. If $|\mathcal{A}|$ is the cardinality of $\mathcal{A}$ and $\mathcal{A}(n)$ the number of unique n-grams over $\mathcal{A}$, then $\mathcal{A}(n) = |\mathcal{A}|^n$. In case of $|\mathcal{A}| = 27$, i.e. the Latin alphabet including the blank character, we obtain 27 possible sub-sequences for uni-grams, already 729 possible sub-sequences for bi-grams and as many as $19,683$ possible sub-sequences for tri-grams. Note that these numbers refer to the hypothetical maximum number of n-grams. In practice, however, the number of distinct n-grams extracted from natural language documents will be considerably smaller than the mathematical upper limit due to the characteristics of the particular language. As an example consider the *tri*-gram *"yyz"*. This *tri*-gram will usually not occur in English or German language documents, except, perhaps, for the reference to the three letter code of Toronto's international airport.

Using character n-grams for describing documents has a number of advantages. First, it is *robust* with respect to spelling errors, second, the token alphabet is known in advance and

is, therefore, *complete*, third, it is topic *independent*, fourth, it is very *efficient* and, finally, it does not require linguistic knowledge and offers a *simple* way of describing documents. Nevertheless, a significant problem is the number of n-grams obtained, if the value of n increases. Most text categorization algorithms are computationally demanding and thus not very well suited for the analysis of very high-dimensional feature spaces. For that reason, it is necessary to reduce the feature space using feature selection metrics.

Cavnar et al. mention in [CT94] a statistical model for describing documents, namely *n-gram frequency profiles*. For each document in the collection, n-grams with different length n are generated. Then, the n-gram occurrences in every document are counted on a per document basis. One objective of this study is to determine the influence of different document representations on the performance of different text-classification approaches. To this end, a character-based n-gram document representation with $n \in \{2, 3\}$ is compared against a document representation based on *word frequencies*. In the word-frequency representation occurrences of each word in a document are counted on a per document basis.

3 Text Categorization

One important task in text categorization is to prepare text in such a way, that it becomes suitable for a text classifier. Generally, the initial number of features extracted from text corpora is very large[1]. Many classifiers are unable to perform their task in a reasonable amount of time if the number of features increases dramatically. Thus, appropriate feature selection strategies must be applied to the corpus. Another problem emerges if the amount of training data in proportion to the number of features is very low. In this particular case, classifiers produce a large number of hypothesis for the training data. This might end up in overfitting [Mit97]. So, it is important to reduce the number of features while retaining those that contain potentially useful information. The idea of feature selection is to score each potential feature according to a feature selection metric and then take the n-top-ranked features. For a recent survey on the performance of different feature selection metrics we refer to [For03].

For this study the *Chi-Squared* feature selection metric is used. The Chi-Squared test is a statistical approach that measures the divergence from the expected distribution with respect to the assumption that the features are independent of the class value. In other words, it evaluates the *worth* of an attribute by computing the value of the chi-squared statistic with respect to the class. Note, that we also evaluated other feature selection metrics such as *Information Gain*. However, since the categorization results are largely the same and due to space restrictions these findings are omitted in this paper.

For the task of document classification, algorithms of three different machine learning areas were selected. The *Naïve Bayes* classification approach, a rule learning approach and support vector machines as a representative of kernel-based learning were applied.

[1] In the following exposition, we will often use the terminology of the machine learning arena. Thus, a *feature* refers to a *term* or an *n-gram* in the document representation. An *instance* refers to a particular *document*.

The *Naïve Bayes* classification approach is based on a probability model which can be derived from the *Bayes' Theorem* combined with the (naïve) presumption of conditional independence. The probability model for the classifier is a conditional model $P(C|E_1, \ldots, E_n)$ over a dependent class variable C. The set C contains a finite number of classes, conditional on the feature variables $E_1, \ldots, E_n$. After applying the *Bayes' Theorem* Equation 1 is obtained.

$$P(C|E_1, \ldots, E_n) = \frac{P(C)P(E_1, \ldots, E_n|C)}{P(E_1, \ldots, E_n)} \tag{1}$$

The denominator of this fraction can be ignored, since it does not depend on C and the values of the features E_i are given. Hence, it can be regarded as constant. The numerator is equivalent to the joint probability model $P(C, E_1, \ldots, E_n)$. Assuming that each feature E_i is conditionally independent of every other feature, it is possible to express the conditional distribution over the class variable C as

$$P(C|E_1, \ldots, E_n) = Z \cdot P(C) \prod_{i=1}^{n} P(E_i|C) \tag{2}$$

where Z is a scaling factor dependent only on $E_1, \ldots, E_i$, i.e., a constant if the values of the feature variables are known. At this point we have derived the Naïve Bayes probability model. Next, the model has to be combined with a decision rule to obtain the classifier. Selecting the hypothesis that is most probable is known as the *maximum a posteriori* or *MAP* decision rule. The corresponding classifier is the function cl, defined as follows

$$cl(e_1, \ldots, e_n) = \mathsf{argmax}_c P(C = c) \prod_{i=1}^{n} P(E_i = e_i|C = c) \tag{3}$$

One way to estimate the parameters of the probability model is to simply use the frequencies observed in the training set. However, as mentioned before the Naïve Bayes classifier assumes that all features of instances in the training set are independent. Although the assumption of independence is questionable in most real-world tasks, Naïve Bayes often performs very well in classification. As McCallum et al. point out in [MN98], this can be explained by the fact that classification estimation is only a function of the sign (in binary cases) of the function estimation; the function approximation can still be poor while classification accuracy remains high. The interested reader is pointed to [DP97] for a detailed evaluation of the Naïve Bayesian classifier.

A rule-learner tries to induce a set of rules for a collection of training data. These rules are then applied on the test collection for classification purposes. Two well-known members of the family of rule-learners are C4.5 [Qui93] and RIPPER [Coh95]. Both approaches perform two steps to induce their rule sets: First, an initial rule set is determined and, second, these rules are discarded or adjusted via a global optimization strategy.

Frank et al. describe in [FW98] a rule-induction approach without the need for applying a global optimization strategy to generate appropriate rules. PART (*Partial Decision Trees*)

adapts the divide-and-conquer strategy of RIPPER and combines it with the decision tree approach of C4.5. More precisely, PART generates a set of rules according to the divide-and-conquer strategy, removes all instances from the training collection that are covered by this rule and proceeds recursively until no instance is left. To generate a single rule, PART builds a partial decision tree for the current set of instances and chooses the leaf with the largest coverage as the new rule. Afterwards, the partial decision tree is discarded. The advantage of this method is the avoidance of early generalization.

A *Support Vector Machine* (SVM) is a learning algorithm that performs binary classification (pattern recognition) and real value function approximation (regression estimation) tasks. The idea is to non-linearly map the n-dimensional input space into a high-dimensional feature space. This high-dimensional feature space is classified by constructing a linear classifier. The basic SVM creates a maximum-margin hyperplane that lies in this transformed input space. Consider a training set consisting of labelled instances: A maximum-margin hyperplane splits the training instances in such a way that the distance from the closest instances (i.e. the margin) to the hyperplane is maximized. For a comprehensive exposition of kernel-based learning methods and Support Vector Machines we refer to [Bur98, MMR$^+$01].

For the study presented herein, the *Sequential Minimal Optimization* (SMO) training algorithm for Support Vector Machines is used. During the training process of a SVM the solution of a very large quadratic programming optimization problem has to be found. The greater the number of instances which constitute a training set gets, the more time and resource consuming the calculation process becomes. For a detailed report on the functionality of the SMO training algorithm for SVMs we refer to [Pla99].

4 Experiments

The major objective of the experiments presented in the remainder of this paper is to compare the performance of different text classification approaches for multi-class categorization when applied to a "noisy" domain. By nature, e-mail messages are short documents containing misspellings, special characters and abbreviations. For that reason, e-mail messages constitute perfect candidates to evaluate this objective. Not to mention the varying length of e-mail messages which entails an additional challenge for text classification algorithms. Moreover, the impact on performance is assessed when header information contained in e-mail messages is taken into account. Hence, two different representations of the corpus are generated to evaluate this issue. Note that all experiments were performed with 10-fold cross validation to reduce the likelihood of overfitting to the training set.

4.1 Data

The document collection consists of 1,811 e-mail messages. These messages have been collected during a period of four months commencing with October 2002 until January

2003. The e-mails have been received by a single e-mail user account at the *Institut für Softwaretechnik*, Vienna University of Technology, Austria. Beside the "noisiness" of the corpus, it contains messages of different languages as well. Multi-linguality introduces yet another challenge for text classification.

At first, messages containing confidential information were removed from the corpus. Next, the corpus was manually classified according to the categories outlined in Table 1. Some of the introduced classes might give the impression of a more or less artificial separation. Introducing similar classes was intentionally done for assessing the performance of classifiers on closely related topics. Consider, for example, the position class which comprises 66 messages mainly posted via the dbworld and seworld mailinglists. In particular, it contains 38 dbworld messages, 23 seworld, 1 isaus messages, and 4 messages from sources not otherwise categorized. In contrast to standard dbworld or seworld messages, position messages deal with academic job announcements rather that scientific conferences and alike. But they still contain the same header and signature information as messages of the dbworld or seworld classes. Hence, the difference between these classes is determined by the message content only.

Next, two representations of each message were generated. The first representation consists of the data contained in the e-mail message, i.e. the complete header as well as the body. However, the e-mail header was not treated in a special way. All non-Latin characters, apart from the blank character, were discarded. Thus, all HTML-tags remain part of this representation. Henceforth, we refer to this representation as *complete* set. Furthermore, a second representation retaining only the data contained in the body of the e-mail message was generated. In addition, HTML-tags were discarded, too. Henceforth, we refer to this representation as *cleaned* set. Due to the fact, that some of the e-mail messages contained no textual data in the body besides HTML-tags and other special characters, the corpus of the *cleaned* set consists of less messages than the *complete* set. To provide the total figures, the *complete* set consists of $1,811$ e-mails whereas the *cleaned* set is constituted by $1,692$ e-mails (cf. Table 1). Subsequently, both representations were translated to lower case characters.

Starting from these two message representations, the statistical models are built. For each message in both sets a character n-gram frequency representation with $n \in \{2, 3\}$ was generated. For the *complete* set we obtained $20,413$ distinct features and for the *cleaned* set $16,362$. Next, we generated the word frequency representation for each set and obtained $32,240$ features for the *complete* set and $20,749$ features for the *cleaned* set. In order to test the performance of text classifiers with respect to the number of features, we subsequently selected the top-ranked n features with $n \in \{100, 200, 300, 400, 500, 1000, 2000\}$ determined by the Chi-Squared feature selection metric.

4.2 Results

Table 2 gives a comparison of the classification results for each category using the character n-gram representation and the word frequency representation. In this case, the classi-

category	label	# complete	# cleaned	description
admin	(1)	32	32	administration issues
dbworld	(2)	260	259	mailinglist
department	(3)	30	29	department issues
dilbert	(4)	70	70	"daily dilbert"
ec3	(5)	20	19	project related messages
isaus	(6)	24	22	mailinglist
kddnuggets	(7)	6	6	mailinglist
lectures	(8)	315	296	lecturing issues
michael	(9)	27	25	no specific topic
misc	(10)	69	67	no specific topic
paper	(11)	15	14	publications
position	(12)	66	66	job announcements
seworld	(13)	132	132	mailinglist
spam	(14)	701	611	spam messages
talks	(15)	13	13	talk announcements
technews	(16)	31	31	mailinglist
totals		**1,811**	**1,692**	

Table 1: Documents of the corpus on a per category basis.

fiers were applied to the *cleaned* set of messages described by 1000 features. The precision (P)[2] and recall (R)[3] values of each individual classifier are depicted.

Note that *NBm* in the caption of Table 2 refers to the multi-nominal Naïve Bayes classifier, *PART* refers to the partial decision tree classifier and *SMO* refers to the Support Vector Machine using the SMO training algorithm. We used the implementation of the learning algorithms as provided with the WEKA machine learning toolkit [WF00]. All classifiers show high precision and recall values when applied to mailinglist classes such as dbworld[4] regardless of their document representation. If *SMO* is applied to the character n-gram representation it outperforms the word frequency representation of the seworld-class. In contrast, the *SMO* classifier has severe problems when it comes to categorizing messages of the michael-class, when based on the character n-grams. In this case, the precision and recall values are zero. *NBm* shows high precision and recall values for the position class, which is closely related to dbworld. However, we obtain lower values for classes such as misc or ec3 which deal with unspecific topics. Studying the confusion matrices showed that particular classes are confused easier than others, for instance misc and lectures are often confused. The classifiers show very high values for the character n-gram representation of the technews class. Almost all precision and recall values are 1, except the *NBm* precision value. The kddnuggets class is correctly classified by all classifiers in the word frequency representation. *SMO* is the only classifier that accomplishes this task with the same results for the character n-grams. For spam filtering, *PART* and *SMO* outperform *NBm* regardless of using character n-grams or word frequencies. The precision and recall values obtained for spam classification are still high, despite sorting messages into 16 classes. Especially, precision is notably high for the spam class, which means that only a tiny number of legitimate messages have been misclassified as being spam.

[2] $\mathrm{P} = \frac{\text{number of relevant documents retrieved}}{\text{total number of documents retrieved}}$

[3] $\mathrm{R} = \frac{\text{number of relevant documents retrieved}}{\text{total number of relevant documents}}$

[4] Due to space restrictions we use the labels as given in Table 1 instead of the category descriptions.

	character n-grams						word frequencies					
	NBm		PART		SMO		NBm		PART		SMO	
Class	P	R	P	R	P	R	P	R	P	R	P	R
(1)	0.44	0.94	0.49	0.59	0.75	0.38	0.67	0.81	0.56	0.59	0.77	0.63
(2)	0.94	0.87	0.94	0.96	0.93	0.95	0.96	0.95	0.92	0.96	0.95	0.96
(3)	0.93	0.97	0.95	0.97	0.96	1	0.97	0.99	0.96	1	0.97	1
(4)	0.60	0.32	0.33	0.26	0.73	0.42	0.91	0.53	0.78	0.74	0.85	0.58
(5)	0.73	0.55	0.35	0.31	1	0.17	0.93	0.48	0.66	0.66	0.88	0.48
(6)	0.47	0.82	0.59	0.59	0.83	0.68	0.82	0.82	0.71	0.68	0.94	0.73
(7)	0.86	1	0.60	1	1	1	1	1	1	1	1	1
(8)	0.73	0.92	0.83	0.80	0.71	0.99	0.68	0.97	0.89	0.87	0.83	0.92
(9)	0.48	0.40	0.68	0.68	0	0	1	0.16	0.96	0.96	0.79	0.76
(10)	0.14	0.05	0.40	0.34	0.57	0.15	0.35	0.15	0.42	0.37	0.66	0.31
(11)	0.82	0.64	0.73	0.57	0.88	0.50	0.82	0.64	0.53	0.64	0.88	0.50
(12)	0.69	0.94	0.65	0.62	0.87	0.70	0.85	0.94	0.77	0.62	0.86	0.73
(13)	0.85	0.93	0.92	0.91	0.96	0.99	0.91	0.98	0.93	0.96	0.94	0.98
(14)	0.98	0.83	0.93	0.94	0.94	0.96	0.97	0.87	0.94	0.96	0.91	0.97
(15)	0.86	1	1	1	1	1	0.97	1	0.97	1	0.97	1
(16)	0.88	0.54	0.42	0.39	0.69	0.69	0.75	0.46	0.55	0.46	0.91	0.77
avg	0.71	0.73	0.67	0.67	0.80	0.66	0.85	0.73	0.79	0.78	0.88	0.77
stdev	0.23	0.29	0.23	0.27	0.25	0.35	0.17	0.29	0.19	0.21	0.09	0.22

Table 2: *Cleaned* set: Precision and recall for each class per classifier.

If the classifiers are applied to the *complete* set, the results depicted in Table 3 are obtained. Again, 1000 features were used. In analogy to Table 2 precision and recall values are given for each class. *PART* shows very high values for the dbworld class when using character n-grams. In fact, it outperforms all other classifiers. The kddnuggets class is identified correctly by almost all classifiers, except *PART*'s low precision value when word frequencies are used. In the case of character n-grams, *NBm* shows very low precision and recall values for talks and misc classes. However, the values for the misc class are low, regardless of classifier and representation. Classification of spam-messages is accomplished best by *SMO* when word frequencies are used. Again, precision values are very high irrespective of the closen categorization approach.

In Figure 1 the classification accuracy[5] of the text classifiers, along the number of features, is shown. In this case, the *cleaned* set is evaluated. Figure 1(a) shows the percentage of correctly classified instances using character n-grams and Figure 1(b) depicts the results for word frequencies. Each curve corresponds to one classifier. If we consider the character n-gram representation (cf. Figure 1(a)) *NBm* shows the lowest accuracy. It starts with 69.2% (100 features), increases strongly for 300 features (78.0%) and arrives at 82.7% for the maximum number of features. *PART* classifies 78.3% of the instances correctly when 100 features are used, which is higher than the 76.7% achieved with the *SMO* classifier. However, as the number of features increases to 300, the *SMO* classifier gets ahead of *PART* and arrives finally at 91.0% correctly classified instances (*PART*, 86.1%). Hence, as long as the number of features is smaller than 500, either PART or SMO yield high classification results. As the number of features increases, *SMO* outperforms *NBm* and *PART* dramatically. In case of word frequencies, a similar trend can be observed but the

[5] $\text{Accuracy} = \frac{\text{number of correctly classified documents}}{\text{total number of documents}}$

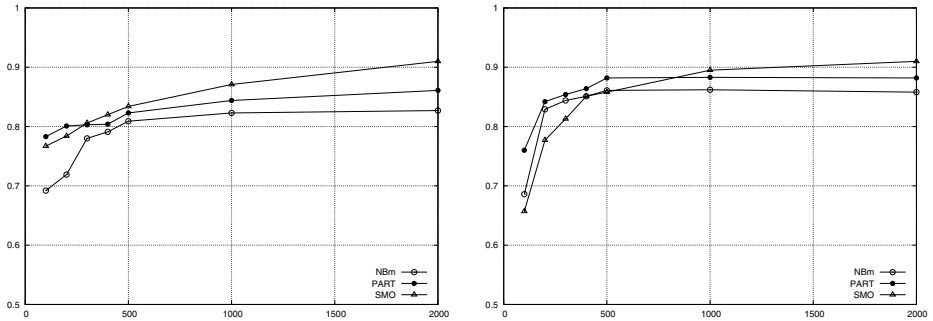
	character n-grams						word frequencies					
	NBm		**PART**		**SMO**		**NBm**		**PART**		**SMO**	
Class	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
(1)	0.31	0.88	0.56	0.59	0.66	0.78	0.54	0.78	0.58	0.59	0.74	0.78
(2)	0.95	0.87	0.99	0.97	0.98	0.96	0.99	0.96	0.97	0.97	0.99	0.97
(3)	0.94	0.96	0.97	1	0.96	1	0.97	0.96	0.97	1	0.97	1
(4)	0.26	0.55	0.67	0.70	0.91	0.50	0.88	0.70	0.75	0.75	0.94	0.85
(5)	0.11	0.97	0.45	0.47	1	0.17	0.95	0.60	0.61	0.57	0.89	0.53
(6)	0.38	0.96	0.92	0.92	0.96	1	0.63	1	0.96	1	0.96	1
(7)	1	1	1	1	1	1	1	1	0.75	1	1	1
(8)	0.46	0.19	0.88	0.83	0.80	0.96	0.51	0.97	0.88	0.87	0.85	0.98
(9)	0.77	0.96	0.92	0.85	0.93	0.96	0.96	1	0.96	1	0.96	1
(10)	0.02	0.04	0.42	0.39	0.70	0.23	0.53	0.30	0.45	0.38	0.67	0.29
(11)	0.19	0.80	0.46	0.40	0.89	0.53	0.90	0.60	0.56	0.60	0.78	0.47
(12)	0.55	0.55	0.80	0.85	0.88	0.80	0.86	0.83	0.87	0.83	0.92	0.88
(13)	0.82	0.99	0.95	0.93	0.94	0.99	0.89	0.99	0.93	0.93	0.96	0.99
(14)	1	0.52	0.97	0.95	0.95	0.98	0.99	0.61	0.95	0.96	0.96	0.98
(15)	0.84	1	1	1	0.97	1	0.72	1	0.97	1	0.97	1
(16)	0.05	0.08	0.80	0.92	0.88	0.54	1	0.85	1	0.92	1	0.92
avg	0.54	0.71	0.79	0.80	0.90	0.78	0.83	0.82	0.82	0.84	0.91	0.85
stdev	0.36	0.34	0.21	0.22	0.10	0.29	0.18	0.21	0.18	0.20	0.10	0.22

Table 3: *Complete* set: Precision and recall for each class per classifier.

roles have changed, cf. Figure 1(b). All classifiers start with rather low accuracies. Remarkably, *SMO* (65.7%) classifies less instances correctly than *PART* (76.0%) and *NBm* (68.6%). All three classifiers boost their classification results enormously, as the number of features increases to 200. At last, the *SMO* classifier yields 91.0% and outperforms both *NBm* (85.8%) and *PART* (88.2%). Furthermore, once the number of features exceeds 500 both *NBm* and *PART* increase their classification results only marginally if at all.

Figure 2 shows the classification accuracy when the *complete* set is used for the classification task. Again, the left chart (cf. Figure 2(a)) represents the percentage of correctly classified instances for character n-grams and Figure 2(b) depicts the results for the word frequencies. If *NBm* is applied to character n-grams, the classification task ends up in an almost random sorting of instances. The best result is achieved when 100 features are used (64.8%). As the number of features grows, *NBm*'s accuracy drops to its low of 54.2% (400 features) arriving at 62.7% for 2000 features. Contrarily, *PART* classifies 84.6% of the instances correctly using 100 features. However, increasing the number of features improves the classification accuracy of *PART* only marginally (2000 attributes, 89.1%). *SMO* starts at 76.1%, increases significantly as 200 features are used (82.8%) and, classifies 92.9% of the instances correctly as the maximum number of features is reached.

In analogy to the results obtained with character n-grams, *NBm* shows poor classification accuracy when word frequencies are used, cf. Figure 2(b). Its highest accuracy is 83.5% as the maximum number of features is reached. Interestingly, *PART* classifies 87.0% of instances correctly straight away. This represents the highest of all values obtained with 100 features. However, *PART*s performance increases only marginally for larger number of features and reaches, at last, 90.9%. *SMO* starts between *NBm* and *PART* with 80.1%. Once 400 features are used, *SMO* moves to first place with 90.8% and arrives at the peak result of 93.6% correctly classified instances when 2000 features are used.



(a) character n-grams

(b) word frequencies

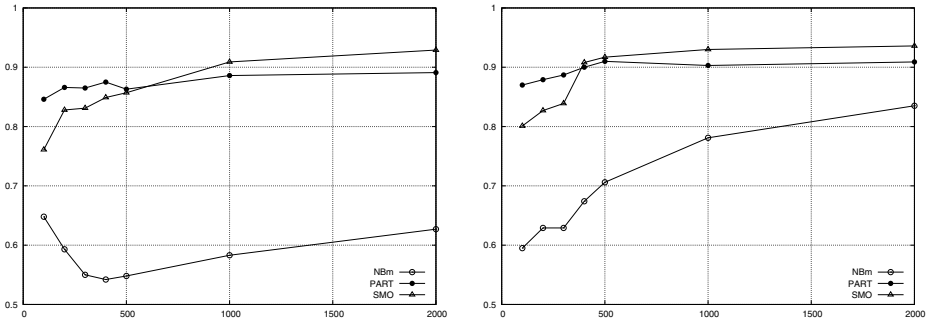
Figure 1: *Cleaned* set: Classification performance of individual classifiers (x–axis: number of features; y–axis: classification accuracy).

5 Conclusion

In this paper, the results of three text categorization algorithms are described in a multi-class categorization setting. The algorithms are applied to character n-gram frequency statistics and a word frequency based document representation. A corpus consisting of multi-lingual e-mail messages which were manually split into multiple classes was used. Furthermore, the impact of e-mail meta-information on classification performance was assessed.

The classifiers, especially *SMO* and *PART*, showed similar classification accuracy regardless of the chosen document representation. However, when applied to word frequencies marginally better results were obtained for all categorization algorithms. Moreover, when a word-based document representation was used the percentage of correctly classified instances was higher in case of a small number of features. Using the word-frequency representation results in a minor improvement of precision and recall. The results, especially those of *SMO*, showed that both document representations are feasible in multi-class e-mail categorization. *PART* revealed its strength as long as the number of features was smaller than 500, regardless of which document representation was used. It was the most stable of all classifiers with respect to the number of features. More precisely, *PART* showed high classification accuracy straight away but improved only marginally as the number of features increased. Eventually, as the number of features increased, *SMO* outperformed *PART* and *NBm* in all cases.

Although a comparison between values obtained for the *cleaned* and *complete* set has to be handled with care as the two representations contain a different number of messages, some speculation can be made. It seems, that the use of the *complete* set slightly narrows the gap between the two document representations' classification accuracy and precision and recall values. *PART*, for instance, shows equal or higher values for precision and recall



(a) character n-grams

(b) word frequencies

Figure 2: *Complete* set: Classification performance of individual classifiers (x–axis: number of features; y–axis: classification accuracy).

in 5-classes compared to 2 for the *cleaned* representation. Interestingly, *NBm* and *SMO* do not increase this number. The inclusion of header information prevents *NBm* from finding a proper model for classification. *PART* increased its performance *dramatically* when the *complete* set was used. In case of character n-grams 84.6% of the instances (100 features) were correctly classified compared to 78.3% for the *cleaned* set. Considering the word frequencies the boost was even higher. A classification accuracy of 76.0% (100 features) was observed using the *cleaned* set. *PART*'s accuracy for the *complete* set was 87.0%, this is the highest of all values obtained for 100 features. Similarly, the number of correctly classified instances increases for *SMO* when the *complete* set was used. This is an indication in favor of our assumption that included header information, although not treated in a special way, has positive impact on the number of correctly classified instances.

Despite the good results obtained for multi-class e-mail categorization, a semi-automatic classification approach, which adapts classes to new instances when e-mails are received, might improve results further. Finally, we assume that sorting e-mail messages into distinct classes is not the only possible way for arranging e-mails. Capturing their semantics and automatically arrange e-mails according to their semantic relationship appears to an alternative approach worth considerating. To this end, we performed some preliminary experiments on automatically arranging e-mails using *Self-Organizing Maps*. The relationships obtained look promising and this approach will be investigated further.

Acknowledgments

Many thanks are due to the Machine Learning Group at The University of Waikato for their superb WEKA toolkit (http://www.cs.waikato.ac.nz/ml/).

The contribution of Monika Köhle is partially based on research performed by Nik Michalopulos from the *Institut für Kommunikationstechnologie, A–1040 Vienna, Austria*.

References

[APK+00] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In *Proc. Workshop on Machine Learning and Textual Information Access, PKDD*, 2000.

[Bur98] C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[Coh95] W. W. Cohen. Fast effective rule induction. In *Proc. Int'l Conf. on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[CT94] W. B. Cavnar and J. M. Trenkle. N-gram-based text categorization. In *Proc. Int'l Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, Las Vegas, NV, 1994.

[DP97] P. Domingos and M. Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2-3), 1997.

[For03] G. Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[FW98] E. Frank and I. H. Witten. Generating Accurate Rule Sets Without Global Optimization. In *Proc. Int'l Conf. on Machine Learning*, pages 144–151. Morgan Kaufmann, 1998.

[Mit97] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[MMR+01] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An Introduction to Kernel-Based Learning Algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, March 2001.

[MN98] A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *Proc. AAAI-98 Workshop on "Learning for Text Categorization"*. AAAI Press, 1998.

[Pla99] J. Platt. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods–Support Vector Learning*, pages 185–208. MIT Press, 1999.

[PS03] F. Peng and D. Schuurmans. Combining naive Bayes and n-gram language models for text classification. In *Proc. European Conference on Information Retrieval Research (ECIR-03)*, pages 335–350, 2003.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[Seb02] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[WF00] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 2000.