Feature Selection and Representation in Text Classification

Henry Robinson

April 22, 2003

Word Count: 5112 excluding bibliography

1 Introduction

Text classification remains an important practical application of both modern machine learning (ML) and natural language processing (NLP) techniques. The influence of these disparate areas of research has contributed much to the success of current state of the art classification methods. This essay provides an overview of the field of text classification, and investigates in particular the topic of *feature representation*, which is concerned with how a characterization of the document to be classified is obtained. It is here that both ML and NLP can play a large role, and this essay argues that encouraging collaboration between the two will result in superior classification performance.

This essay is split into three sections. The first gives a broad introduction to text classification, including a formal statement of the problem after [1]. Text classification is motivated here, to provide a practical context for the ensuing mainly theoretical discussion. The second section details popular approaches to text classification, including details of common experimental procedures for evaluating the effectiveness of a new technique. Finally, the third section concentrates on representational issues, and surveys some of the literature associated with the topic.

1.1 What is text classification?

Text classification is the process of assigning labels to pieces of text, i.e. giving each document one or more *categories*. These categories may or may not be exclusive, that is a document may or may not belong to more than category. Indeed, a document may not belong to any categories at all. These issues are typically domain specific.

Clearly, there is a need to strive for ever increasing accuracy, but all the time being careful not to compromise classification rate. This is the classic engineering tension of, essentially, quality vs. quantity. The vast majority of current research concentrates on improving classification accuracy, since with the ever increasing power of commodity computer hardware the performance is taking care of itself.

Formally, text classification involves populating a decision matrix $D = \{d_{ij}\}$ with values from $\{0, 1\}$. Each cell d_{ij} contains a 1 if document i is assigned category j, and a 0 otherwise. The decision matrix is described in[1]. I suggest generalising the binary matrix by allowing d_{ij} to take on any normalised real value rather than one from $\{0, 1\}$. This allows the classifier to express a degree of *confidence* in its own decisions, which may be useful for referring problem cases to a human to make the final decision, for example. We can move from this slightly altered decision matrix D' suggested here to the binary matrix D simply by either applying some threshold τ such that $d_{ij} = 1$ if $d'_{ij} \geq \tau$ and 0 otherwise, or simply setting the n highest confidence values in each row d_i to 1 and the rest to 0. Setting $n = 1$ ensures that at most one category is assigned to a document.

1.2 Applications

There are very many potential applications of text classification techniques. In this section I examine some of them.

1.2.1 Spam filtering

Close to many readers' hearts will be the persistent problem of *spam*, that is the receipt of vast quantities of unsolicited junk e-mail. A text classification system could, in the ideal case, categorize incoming messages into genuine and spam categories, rejecting those that it found to be spam. Such systems are already at large, but, since they cannot guarantee 100% success, still must accept all the spam messages for the user to review at a later date (since the cost of rejecting a valid e-mail is typically far greater than the cost of receiving 100 spam mails). There are several recent papers devoted to this topic [17][19].

1.2.2 Summarisation Evaluation

An interesting application of text classification is the ability to evaluate automatically produced summaries of text [11]. Working on the assumption that a summary should capture the significant features in a document, it may be argued that a summary should classify the same as the full length document. This idea may be turned around to suggest that summaries may be automatically generated from feature vectors.

1.2.3 Newsgroup Sorting

Another common gripe of frequent internet users is the proliferation of 'off-topic' comments posted to newsgroups and bulletin boards. These mistakes are often the consequence of the posters' unfamiliarity with the topic segregation rather than any malicious intent. A TC system could be employed to recommend the correct recipient for a user's message.

1.2.4 Document Organisation

A news or media company will typically see hundreds of submissions a day. In order to efficiently handle such a vast flow of information, an automatic TC system would classify each document by topic so that they could be piped to the relevant recipient.

1.2.5 Hyperline Traversal Predication

Mladenic [2] uses a TC system to predict which hyperlinks on a given web-page the user is likely to click on. Each hyperlink text description is treated as a miniature document. While there are doubtless other psychological factors at work (such as visual appearence of the hyperlink, position on the page, relevance of the hyperlink to some desire of the user), a TC system could be used to naively predict the next page for a fast look-ahead caching system.

2 Text Classification Approaches

Historically, the first attempts at automated classification involved the manual generation of deterministic rules that assigned a document to a category. This approach was fraught with problems, amongst them the chronic lack of scalability of the rule set and maintenance issues required to deal with generalising to new documents and categories. In short, the rule approach required too much work for too little gain. As such, a move to a less maintenance heavy formalism was required.

Modern classification approaches now employ Machine Learning techniques. ML is a fast growing field of computer science which is concerned with the creation of computer programs that learn from experience [14]. In the case of text classification, the task to be learnt is an *objective function* from a set of functions called the *hypothesis space*, which maps candidate documents onto one or more categories.

A set of pre-classified documents provides the experience necessary for a classifier to learn the objective function. This set is typically marked up by a human, and is the only human input required to operate a classifier: the learning and subsequent classification can be done automatically. It is possible to learn from

a document set that has not already been classified (this is *unsupervised learning* in contrast to *supervised learning*), but the vast majority of approaches do not consider this possibility. Classification performance is typically significantly higher with supervised learning.

2.1 Classifiers

Research in ML has resulted in many different classification schemes. These are typically characterised both by their candidate hypothesis space, from which the objective function is drawn, and their learning method which selects the objective function. Following is a description of some of the most popular classifiers (naive Bayes omitted for space reasons).

2.1.1 k-Nearest Neighbours

Perhaps the simplest classification scheme is *k Nearest Neighbours* or kNN. Working on the assumption that training data close in feature vector space to the point to be classified will have the same classification as the test point, the algorithm examines the k closest points to the test point and classifies the test point the same as the majority of the k nearest neighbour points. The training algorithm therefore does nothing, and the classification algorithm simply examines the nearest k points to the test point. In the case of a tie, the class with a point closest to the test point is chosen.

Galavotti et. al. [8] extend the kNN classifier to makes use of negative evidence: nearby points that do not belong to the class in consideration subtract from the similarity score, and only those classes with a positive similarity score are chosen. This 'kNN_{neg}' classifier was found under certain conditions to be useful, but less robust to the choice of k than a straightforward kNN classifier.

2.1.2 Support Vector Machines

Support Vector Machines are linear classifiers that construct a hyperplane in some high-dimensional space and classify points according to which side of the hyperplane they lie on. However, most feature spaces are not *linearly seperable*, that is a hyperplane that neatly divides the space into positive and negative examples typically does not exist. SVMs solve this problem by projecting the feature space non-linearly into an inner-product space which is linearly separable. This is accomplished by means of a *kernel function*, and SVMs are thus one instance of a set of classifiers called kernel methods.

2.1.3 Rocchio Classifiers

Rocchio classifiers assume that the representation of a particular category must combine the properties of both positive and negative example documents

[10]. The training algorithm consists of constructing a representative vector w_j for each category, which is updated according to

$$w'_j = \alpha w_j + \frac{\beta}{|C|} \sum_{i \in C} v_{ij} - \frac{\gamma}{n - |C|} \sum_{i \notin C} v_{ij}$$

where n is the total number of documents, C is the set of documents assigned to the topic under consideration, v_i is the feature vector constructed from training document i and α, β and γ are constants.

Once these representative vectors are constructed, classification involves comparing the input feature vector against each of them in turn, according to some similarity measure such as a cosine measure or Euclidean distance.

2.2 Experimental Issues

2.2.1 Choice of corpora

There are many corpora available for development of TC systems. Perhaps most popular is the Reuters dataset, which consists of a large selection of newswire items, each marked with a selection of topic codes. In order for a system to be fairly trained, any corpus must be divided into training and test data, and neither data set must be used for the other's purpose. Typically there are many more training data than there are test data.

Issues of data sparsity are rarely, if ever, mentioned in the literature, which implies that in general there is assumed to be 'enough' data. This is perhaps a dangerous assumption, and it would be interesting to see what effect data smoothing techniques had on the performance of the classifiers. However, obvious data smoothing schemes would make it difficult to calculate many of the information measures given in section 3, since term and category co-occurence information is needed to populate the contingency table.

2.2.2 Evaluation

TC systems are typically evaluated through the well known precision and recall measures, where precision is the ratio of correct classifications to classifications made, and recall is the ratio of correct classifications to total classifications in the test data. These may be averaged over an 11 point scale, as in [13].

3 Feature Selection and Representation

As seen in section 2, almost every popular classifier accepts as input a feature vector that characterises the document to be classified, as well as those that were trained on. Clearly, the construction of this feature vector is very important to the successful operation of the classifier. There are essentially three separate

problems to be solved in feature vector construction. The following describes these problems, and gives an overview of the literature relevant to each.

3.1 Feature Extraction

Feature extraction concerns the identification of the features that will be considered for inclusion in the feature vector. The majority of approaches described in the literature use a simple 'bag of terms' approach to feature extraction that includes all the words in a document modulo a *stop list*, a list of the most common words that are unlikely to be distinguishing features.

It may be argued, however, that a simple bag of terms representation is not sufficiently representative of the aspects of the document that we wish to capture. For example, a TC system that attempts to classify text according to author will not only need to consider the vocabulary of the writer (represented by the bag of terms), but also more structured linguistic features of the text, such as the frequency of passivised verbs, the number of fragment sentences etc. These features will likely give a greater clue to the identity of an author than the words used (although these are also important).

Basili et. al. [12] make an attempt at introducing linguistically motivated feature extraction by including both proper nouns and what they call 'Terminological Expressions'. They see a small improvement in classification performance, but only over a simple baseline system. There are some other applications of simple linguistic knowledge (for example, Caropreso et. al. [4] investigate the use of statistically selected n-grams, with moderate success), but there is little evidence in the literature at truly NLP motivated attempts at feature extraction.

3.2 Feature Selection

The second problem to be solved is *feature selection*. The number of candidate features identified by feature extraction may be extremely large (in the bag of terms case, the size of the feature vector is approximately linear in the length of the document). This has a serious impact on the behaviour of the classifier. If all the features are statistically independent, then there are theoretical arguments that more features should lead to better classification performance [15]. However, when faced with limited training data, adding more features may actually *decrease* classification performance since the data is too sparse to learn to distinguish on so many features. This is called the *curse of dimensionality*, and is best solved by reducing the dimensionality of the feature vector.

There is an efficiency argument for reducing the feature vector size as well. The performance of many classifiers is directly dependent on the length of the feature vector - for example, the distance calculations required in a kNN classifier scale linearly in computational cost with the number of features. Naive Bayesian

classifiers become computationally intractable when the feature vector gets too large.

Hence feature selection, which aims to reduce the dimensionality of the feature vector by only retaining those features that are most *informative*, or most likely to distinguish. The notion of information content is encapsulated by several theoretical measures [13] [8], as well as more empirical metrics [10] and less direct measures retrieved as part of the learning procedure [3].

3.2.1 Information Measures

An excellent and thorough review of individual feature selection techniques is given in [13]. Starting from a bag of words feature vector, Yang and Pedersen's approach was to score each feature based on some measure of information content, and then remove those features whose information score fell below some threshold. This threshold was adjusted to retain a particular proportion of terms so that different information measures could be compared fairly. This is referred to in the literature as *feature filtering*, but since all feature selection methods can be described as selecting the n best features, the distinction is unnecessary. As is typical, words that appeared in a stop list were removed from the feature vector beforehand.

Five information measures were tested. Document Frequency (DF) is the number of documents in the training corpus in which a term occurs. Information Gain (IG) is a 'measure of the number of bits of information obtained for category prediction by knowing the presence or absence of a term in a document', and is given as:

$$G(t) = H(c) - (P(t)H(c|t) + P(t')H(c|t'))$$

(reformulated for clarity from the version in [13]) where t is the term under consideration, $\{c_i\}_{i=1}^{m}$ is the set of categories and $P(\alpha)$ and $H(\beta)$ are a probability measure and Shannon's entropy measure respectively, which may both be estimated from the training corpus.

The third information measure was Mutual Information (MI). In order to fully explain this, and other information measures, it is helpful to introduce the *contingency table* (see figure 3.2.1). The contingency table records co-occurence statistics for terms and categories. Therefore we see that the number of times a category c occured without the presence of term t in the training corpus was C, for example. We also have that the number of documents, $N = A + B + C + D$. These statistics are very useful for estimating probability values.

	c	c'
t	A	B
t'	C	D

Figure 1: The Contingency Table

The MI measure is given by

$$I(t,c) = \log \frac{P(t\&c)}{P(t)P(c)}$$

which may be approximated by $\frac{A \times N}{(A+C)\times(A+B)}$. Since MI gives values for (category, term) pairs, rather than individual terms, Yang and Pedersen calculate both the maximum and average MI for each term, and test both. This highlights an interesting subtlety in the way that most TC systems operate. Feature selection is performed across *all categories* at once. Basili et. al. [12] argue that feature selection should be performed on a per-category basis to account for different relevance distributions between classes. That is, terms that may be irrelevant with respect to one class may be important distinguishing terms with respect to another. Since many classifiers are binary classifiers (that is, they give *yes* or *no* answers to each category in turn), and therefore a classifier per category is trained, it seems sensible that feature selection should be performed for each category, for maximal performance. This is unlikely to affect Yang and Pedersen's comparative study, but there is little mention of this issue in the literature.

The final two information measures examined in [13] are the χ^2 statistic, given as

$$\chi^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)}$$

(again both average and maximum values were used) and the Term Strength (TS): $s(t) = P(t \in y | t \in x)$ given a document pair (x, y), where x and y are said to be similar according to some similarity measure.

Yang and Pedersen ran tests on a variety of information measure / classifier / corpus permutations, and found that in the best case, feature selection was extremely successful: with 98% of features removed the performance of a kNN classifier was significantly better when using IG than with 100% of the features left intact! This serves to reinforce the importance of keeping the feature vector length manageable for trainability purposes. In general, they concluded that IG, DF and χ^2_{max} performed similarly (and even went so far as to investigate the statistical significance of a correlation between the three, which was quite evident). TS performed well until the feature reduction became more than about 50%, whereas MI performed very poorly indeed. They attribute the

performance of IG and χ^2_{max} to three significant properties of their computation: first, that they give common terms higher relative scores, secondly that they include categorical information and thirdly that they make use of the second row of the contingency table, i.e. they build in negative examples of term presence into their calculation. This last reason connects with a point made by Galavotti et. al. [8], who suggest that building in negative evidence into a classifier will improve performance.

The same paper introduces a scalar measure of the extent of feature selection called the *aggressivity*, which for a full feature set of size r and a reduction to r' is given by $1 - \frac{r'}{r}$. More feature reduction results in a higher aggressivity. Note that this measure is normalised, and is therefore comparable across feature sets. Galavotti et. al. are unclear about the precise relationship between aggressivity and classification accuracy: clearly a high aggressivity gives a low computational cost, but, as Yang and Pedersen showed, a low aggressivity does not necessarily mean better accuracy, and the relationship between the two is in fact quite dependent on both the feature selection technique and the classifier. Hence aggressivity is not very useful as an indicator of the tension between computational cost and classification accuracy.

Galavotti et. al. also introduce a new information measure for feature selection, based on removing counter-intuitive factors from the χ^2 formula. Their proposed simplification, called 'simple χ^2', is given by

$$s\chi^2(t_k, c_i) = P(t_k, c_i)P(t'_k, c'_i) - P(t'_k, c_i)P(t_k, c'_i)$$

which, although they do not mention it in the paper, can be estimated by the determinant of the contingency matrix divided by N. They again find that taking the maximum value $s\chi^2_{max}$ gives the best performance, and discover that $s\chi^2_{max}$ begins to outperform χ^2_{max} at about 98% feature reduction, and DF at about 95%. While they are correct in concluding that this makes $s\chi^2_{max}$ a suitable choice for learning algorithms that scale extremely badly with feature vector size, the classification performance with both a kNN classifier and a Rocchio classifier drops sharply at around these levels, so unless a classifier that is able to make dramatic improvements on the current state of the art with very small feature vectors is discovered, it seems unlikely that $s\chi^2_{max}$ will see any use in anger. Once the feature vector becomes so small, the classification performance of any system must be impaired due to the poor distinguishing power of so few features.

The final information measure considered here is proposed by Basili et. al. [12]. They suggest generalising the Rocchio formula

$$w'_j = \alpha w_j + \frac{\beta}{|C|}\sum_{i \in C} v_{ij} - \frac{\gamma}{n - |C|}\sum_{i \notin C} v_{ij}$$

to feature selection by discarding those features for whom $w_j \leq 0$. While this is an interesting metric, the paper does not do much to justify its selection, nor provide results to support it. Indeed, the greatest success claimed by the authors is a feature reduction of below 50% - hardly comparable with the 98%+ that Yang and Pedersen were achieving! The paper also claims to present an "optimal" method of arriving at values for β and γ, by setting $\beta = 1$ and then varying γ until optimum performance is reached. However, the meaning of 'optimal' is never explained, but certainly does not correspond to an optimal selection of features.

3.2.2 First m Terms

Wibowo and Williams [10] take a dramatically simpler approach to feature selection. Their problem domain is the classification of documents into hierarchical categories. They argue that stylistically, most documents include some statement of their topical intent in the first few sentences, and thus features pertaining to topic classification will be found at the beginning of any document. To test this perhaps tenuous claim, they train several sets of classifiers where feature selection is done by selecting either the first m terms (for $200 \leq m \leq 2000$), or by removing words based on one of two stoplists. They see a near universal improvement of the first m classifiers over the stop-list based ones, and still further improvement over an all-terms classifier, and use this to substantiate their claim that features are 'best' selected for topic classification by using the first m terms. This claim is not without its flaws. A stop-list is normally a given in TC systems, and is always used in conjunction with other feature selection approaches. Wibowo and Williams use the relatively poor performance of the stop-list classifiers as a reason to claim that stop-lists are not as important as the TC literature suggests: however by doing this they are missing the point that it is the interaction of the stop-list with other feature selection techniques that makes it useful. Removing extremely common terms is unlikely to dramatically improve classification performance since the removed terms will not be informative: the only benefits may be for computational efficiency and perhaps trainability of the classifier. However, by removing these terms from consideration by a more informed feature selection technique, a stop-list allows a theoretically rather than empirically motivated selection technique to more precisely identify the informative terms, without the noise that extremely common, empirically uninformative words will contribute.

This leads to the conclusion that Wibowo and Williams have not compared their feature selection technique against the state of the art proposed by Yang and Pederson [13] - despite referencing their paper - which casts doubt upon the usefulness of their results. The fact that they see improvements over all-term classifiers may simply be a result of the aggressivity of their selection, which improves the trainability and sometimes performance as Yang and Pederson showed, rather than as a result of the true informativeness of the first m terms. This is substantiated by the performance decline as m increases.

An interesting experiment would be to consider the in-document distribution of the terms that, say, χ^2_{max} feature selection does not remove. This would serve to either support or dispel Wibowo and Williams' claim about the location of informative terms in the first few sentences of a document.

3.2.3 Summarisation as Feature Selection

Kolcz et. al. [11] offer a novel contribution to feature selection techniques. Arguing that a summary should preserve the important and relevant features of a document, the authors suggest that using the features extracted from a summary will in the ideal case give the same classification performance as feature selection performed on the whole document. This is a slightly flawed assumption: the categories that the document may be assigned to may have very little to do with the information that the summary captures. For example, a classifier intended to sort documents according to the style of the writer (journalistic, academic, popular fiction etc.) would gauge little from a summarisation scheme that presumably attempted to be style neutral. However, with current summarisation technologies and the concentration on topic based classifiers, it seems reasonable to assume that in the general case, a summary should capture the information that a classifier will be most effective with.

To test this claim, the authors compared the effectiveness of feature selection using MI to that of a number of summarisation techniques (none of which were particularly complex - taking either the title, or a paragraph or sentence which scored most highly according to some 'goodness' measure such as number of keywords). They found that in the average case, summarisation techniques performed approximately equivalently to MI feature selection. However, this is another case of the authors not comparing their results against the true state of the art, since Yang and Pedersen found MI performed particularly badly in their experiments (which is in direct contrast to what Kolcz et. al. actually claim Yang and Pedersen found!).

The premise of the paper (the equivalence between summarisation and informative features) is an interesting one, with direct consequences for Information Retrieval. A summary could be entered into a such engine, which would then classify the summary and compare documents in the same categories for relevance - search times could be significantly reduced. However, the refusal of the authors to compare their results to the state of the art, coupled with the naivity of their summarisation techniques, mean that the central premise remains unsubstantiated.

3.2.4 Hybrid Feature Selection

The majority of the feature selection literature considers feature selection techniques in isolation, without considering their interaction with other methods (as mentioned above, stop-lists are an exception to this). Rogati and Yang

[9] tackle this omission by empirically measuring the classification performance of a wide variety of combinations of feature selection techniques. They attempt to find feature selection methods that are high-performing across classifiers and collections (or to empirically disprove their existence), and to find whether combining uncorrelated, high-performing methods yields a performance increase.

Feature selection methods are combined by scoring a term with the maximum normalised score given to it by the selection methods under consideration. This considers all selection methods to be of equal importance: future work should perhaps investigate weighting term scores according to selection method performance. As well as investigating both term frequency and binary weights for each term in the final feature vector, a 'reverse stop-list', or 'cut list' is employed which eliminates rare words (those with a DF ≤ 5). Clearly a domain independent cut-list cannot be prepared in the same manner as a stop-list, hence the reliance on the document frequency of the rare terms.

The results of this experiment showed that combining χ^2_{max} with either DF or IG resulted in the best performance, with a small boost if a cut-list is used. This is consistent across all classifiers, but in the main different classifiers were shown to be widely different in their sensitivity to different feature selection techniques. No attempt is made to analytically justify these conclusions, save for the suggestion that the lack of correlation between χ^2_{max} and DF and IG is responsible, despite the claim in [13] (written by one of the authors of this paper!) that there is a strong correlation between the three information measures.

3.3 Feature Weighting

The third and final component of feature vector construction is *feature weighting*, that is assigning scalar weight values to every feature left in the vector after extraction and selection. Weighting serves to scale the feature vector, which has consequences on similarity measures often used in classifiers. Consider the most often used distance metric, employed by kNN classifiers amongst others, the Euclidean:

$$D(w1, w2) = \sqrt{\sum_{i=1}^{m} (w1_i - w2_i)^2}$$

where w_j is the weight applied to the j^{th} feature. Clearly the distance computation depends heavily on the feature weights.

It may be considered that feature weighting and feature selection are, in essence, equivalent tasks, since we can assign 0 weights to features that we choose not to select. However, it makes sense to treat them as separate endeavours, so that feature weights can be assigned independently of whatever score terms are given to assess their ability to distinguish. That is, feature selection scores

classes of terms relative to other classes of terms, where term weighting weights individual terms relative to other terms in the same class.

Typically, one of two approaches to term weighting are used - binary values (which indicate the presence or absence of a term) and TF*IDF values, which are the product of the term frequency over the training documents and the log inverse document frequency. Rogati and Yang [9] also employ a simple term frequency weighting. However, they empirically show this weighting to perform sub-optimally independently of the feature selection technique, training corpus or classifier.

Given that TF does not outperform simple binary weights, it seems that documents are similar based on whether or not a term appears in both of them, not the relative frequencies of that term. Another approach to term weighting might be to scale the entire feature vector space based on term class scores produced at feature selection time, such that more informative terms contribute more to a notion of similarity than less informative ones. This is not compatible with binary weights (since co-incident points are still co-incident under any scaling), but perhaps small random peturbations of the feature vectors might have the desired effect - these peturbations would be emphasised for uninformative terms under a scale transformation, and thus the distance between those two terms (the $w1_i - w2_i$ component of the Euclidean for a given i) would be larger. To my knowledge, no-one has tried this yet.

4 Conclusions

For all the success that current text classification systems have had, it is clear that more can and should be done. Feature selection techniques are mature and well investigated, thanks to high quality empirical investigations like [13] and [9], as are classification techniques that are in the main domain independent, and subject to substantial theoretical analysis. However, it is the domain dependent techniques such as feature extraction and weighting that require more investigation. There is a substantial body of linguistic theory that should be brought to bear on the question of feature extraction: one central tenet of grammatical theories is that the meaning of a piece of text is not a function simply of the identity of the words in the text, but also their structure. There is little evidence of this being taken into account by the TC community. As mentioned in section 3.1, some work has been done on building in a greater degree of NLP knowledge, but the current attempts do not go far enough, straying little beyond the identification of compound nouns. Once better feature extraction is established, the work already done on selection should come further into its own. Faced with better ideas of the structural importance of features that will result from the application of NLP expertise, feature weighting should also become a more informed technique, rather than the ad-hoc, minimal impact methods currently in use.

References

[1] Fabrizio Sebastiani, *A Tutorial on Automated Text Categorisation*

[2] Dunja Mladenic, *Feature Subset Selection in Text-Learning*

[3] J. Brank, M. Grobelnik, N. Milic-Frayling, D. Mladenic, *Feature Selection Using Support Vector Machines*

[4] Caropreso, Matwin, Sebastiani, *A Learner-Independent Evaluation of the Usefulness of Statistical Phrases for Automated Text Categorization*

[5] Hwee Tou Ng, Wei Boom Goh, Kok Leong Low, *Feature Selection, Perceptron Learning and a Useability Case Study for Text Categorization*

[6] Mladenic, Grobelnik, *Feature Selection for Unbalanced Class Distribution and Naive Bayes*

[7] Chakrabarti, Dom, Agrawai, Raghavan, *Scalable Feature Selection, Classification and Signature Generation for Organizing Large Text Databases into Hierarchical Topic Taxonomies*

[8] Galavotti, Sebastiani, Simi, *Experiments on the Use of Feature Selection and Negative Evidence in Automated Text Categorization*

[9] Rogati, Yang, *High Performing Feature Selection for Text Classification*

[10] Wibowo, Williams, *Simple and Accurate Feature Selection for Hierarchical Categorisation*

[11] Kolcz, Prabkarmurthi, Kalia, *Summarization as Feature Selection for Text Categorization*

[12] Basili, Moshitti, Pazienza, *A Hybrid Approach to Optimize Feature Selection Process in Text Classification*

[13] Yang, Pederson, *A Comparative Study on Feature Selection in Text Categorization*

[14] Mitchell, *Machine Learning*

[15] Huang, Acero, Hon, *Spoken Language Processing*

[16] Yiming Yang, *Noise Reduction in a Statistical Approach to Text Classification*

[17] Jason D. M. Rennie, *ifile: An Application of Machine Learning to E-Mail Filtering*

[18] Jan Sykes, *The Importance of Indexing*

[19] Mehran Sahami, Susan Dumais, David Heckerman, Eric Horvitz, *A Bayesian Approach to Filtering Junk E-Mail*