Cranfield UNIVERSITY

Decision Engineering Report Series

Edited by Rajkumar Roy and David Baxter

TEXT CLASSIFICATION METHOD REVIEW

By Aigars Mahinovs and Ashutosh Tiwari

April 2007

Cranfield University
Cranfield
Bedfordshire
MK43 OAL
United Kingdom

http://www.cranfield.ac.uk

ISBN 978-1-86194-128-2

'*Decision Engineering*' is an emerging discipline that focuses on developing tools and techniques for informed operational and business decision-making within industry by utilising data and information available at the time (facts) and distributed organisational knowledge.

The '*Decision Engineering Report Series*' from Cranfield University publishes the research results from the ***Decision Engineering Centre***. The research centre aims to establish itself as the leader in applied decision engineering research. The client base of the centre includes: Airbus, BAE SYSTEMS, BOC Edwards, BT Exact, Corus, EDS (Electronic Data Systems), Ford Motor Company, GKN Aerospace, Ministry of Defence (UK MOD), Nissan Technology Centre Europe, Johnson Controls, PRICE Systems, Rolls-Royce, Society of Motor Manufacturers and Traders (SMMT) and XR Associates.

The intention of the report series is to disseminate the centre's findings faster and with greater detail than regular publications. The reports are produced on the core research interests within the centre:

- Cost Engineering
- Product Engineering
- Applied soft computing

Edited by:

Professor Rajkumar Roy
r.roy@cranfield.ac.uk

David Baxter
d.baxter@cranfield.ac.uk

Decision Engineering Centre
Cranfield University
Cranfield
Bedfordshire
MK43 OAL
United Kingdom

http://www.cranfield.ac.uk

Series librarian:

John Harrington
j.harrington@cranfield.ac.uk

Kings Norton Library
Cranfield University

Publisher:

Cranfield University

Abstract

With the explosion of information fuelled by the growth of the World Wide Web it is no longer feasible for a human observer to understand all the data coming in or even classify it into categories. With this growth of information and simultaneous growth of available computing power automatic classification of data, particularly textual data, gains increasingly high importance.

This paper provides a review of generic text classification process, phases of that process and methods being used at each phase. Examples from web page classification and spam classification are provided throughout the text. Principles of operation of four main text classification engines are described – Naïve Bayesian, k Nearest Neighbours, Support Vector Machines and Perceptron Neural Networks. This paper will look through the state of the art in all these phases, take note of methods and algorithms used and of different ways that researchers are trying to reduce computational complexity and improve the precision of text classification process as well as how the text classification is used in practice. The paper is written in a way to avoid extensive use of mathematical formulae in order to be more suited for readers with little or no background in theoretical mathematics.

Keywords: text classification, bayes, kNN, SVM, neural network, feature extraction, feature reduction, web page classification

Table of Contents

1. Terminology

- Internet – worldwide, publicly accessible network of interconnected computer networks two main parts of which are e-mail and Web;
- Web – system of interlinked, hypertext documents or web pages;
- web site – an organisational unit containing a set of web pages that have the same domain name part of their address;
- spam – unsolicited bulk electronic messages or other content that mimics useful content to attract users, but is in fact not useful (and usually machine generated) and usually incorporates some way to capitalise on the attracted users with methods ranging from advertisements to stock price manipulations;
- ham – useful content, opposite of spam;
- spamminess – property of a text describing the probability that the text is spam;
- conversion – a term in marketing describing an actual end result of an advertisement when a potential customer after seeing an advertisement does the purchase of the product advertised;
- true positive – situation in text classification when the classifier correctly classifies a positive test case into the positive class;
- true negative – situation in text classification when the classifier correctly classifies a negative test case into the negative class;
- false positive – situation in text classification when the classifier incorrectly classifies a negative test case into the positive class;
- false negative – situation in text classification when the classifier incorrectly classifies a positive test case into the negative class;
- cybersquatting – practise of a hostile takeover of an Internet domain name either by intercepting the domain registration at renewal or by registering a domain with the same or similar name as a popular product before the owners of the product do so.

2. Introduction and process

Text classification is the act of dividing a set of input documents into two or more classes where each document can be said to belong to one or multiple classes (Sebastiani, 2002). Huge growth of information flows and especially the explosive growth of Internet promoted growth of automated text classification. Development of computer hardware provided enough computing power to allow automated text classification to be used in practical applications. Text classification is commonly used to handle spam emails, classify large text collections into topical categories, manage knowledge and also to help Internet search engines.

The process of text classification as seen from the point of view of automatic text classification systems can be clearly separated into two main phases:

1. information retrieval phase when numerical data is being extracted from the text;
2. main classification phase when an algorithm processes this data to make a decision on what category should the text belong to.

Additional phases are added (Pant & Srinivasan, 2005) to the classification process to reduce the amount of computation and train the algorithms with training data before the actual classification (Fig. 1).

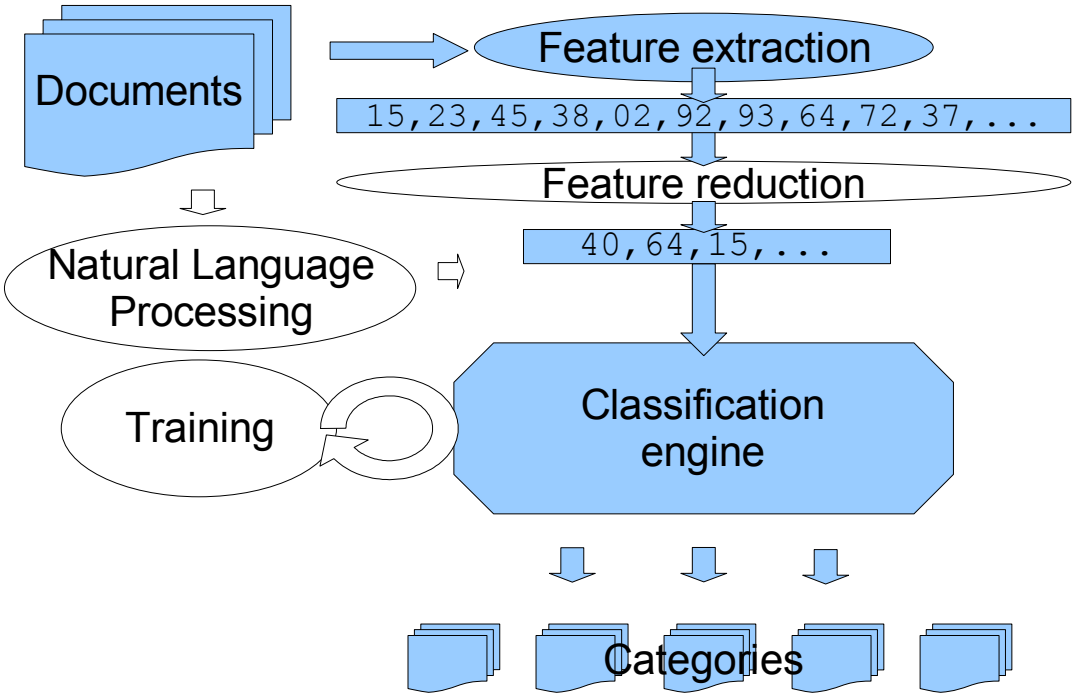

Figure 1: Text classification process diagram

This paper will look through the state of the art in all these phases, take note of methods and algorithms used for reducing computational complexity and improving the precision of text classification process. While many articles (Evgeniy Gabrilovich, 2006) are written on these topic over the years, this paper aims to provide simple and practical overview of the main practical points of the process.

3. Industrial background

This review was sponsored by an Internet marketing company Merjis with an aim to find ways that artificial intelligence technologies and namely text classification could be used to help in the everyday problems in the Internet marketing domain. One of the problems that Merjis was facing was the large number of "spam" websites that place advertisements, from the Google AdSence network or other similar advertisement networks, on their pages. Most of the time on such pages the advertisements are only marginally related to the content of the page and people that click on such ads were observed to have a much lower probability of "converting", that is of purchasing the

advertised goods or services. However, once a user clicks on the advertisement, the site that is being advertised must pay (via Google, in the case of Google AdSence) a fixed fee per click to the site where the advertisement is placed. Therefore, spam sites luring visitors to advertisements of products that these visitors are not intending to purchase are simply a waste of money for the advertiser. Merjis manages advertisement campaigns for many clients and thus is willing to create a system that would be able to identify such spam websites among the hundreds of thousands of websites that participate in the advertisement networks.

Usually text classification engines are tested with datasets where texts must be classified according to their topic. However in the case of the Merjis project, the classifier needed to ignore the topic and go beyond that by trying to evaluate "spaminess" of the site, where "spaminess" can only be defined as overall probability that visitors that get to this site, if they click on our advert, would be interested in purchasing the good or service that this advert is advertising. The measure in many cases is hard to evaluate even for a human expert, leading to fuzzy training data. Such measure can also be abstracted to a more generic "quality" scale.

In addition to an unusual target, the web spam problem exhibits many other complications that are know in the text classification domain. It is common for the available training data to have a high bias towards one end of the scale. In our test case spam websites outnumbered non-spam websites 4 to 1. In such conditions most tested classifiers showed performance that was lower then the baseline performance of a one-way classifier or achieved performance equal to the one-way classifier by simply classifying all test cases into one class. Adjustments to the feature selection algorithms and over-sampling of the training data that were suggested by other researchers had little effect in this particular case. Another issue that had to be taken in consideration in the design of the classifier for the Merjis problem was that a there was a significant difference between the costs for different mistakes that the classifier could make. If classifier would classify a non-spam site as spam, then potential sales were lost, but if spam site was misclassified as non-spam then part of the advertisement budget would be wasted. In the conclusion of the project we could see that none of the used techniques could improve the classification performance enough to achieve break-even on the advertisement investment in the particular case.

If the system would be put further into the production, other challenges would arise – Internet is a very dynamic place. Where now is a thriving site with high quality content, in a couple month could be a spam site set up by a cybersquater. Thus both adding new sites to the existing data set and expiration or reverification of the old sites would present additional challenges.

However the most important problem that this researcher faced in the execution of this industrial project was the lack of a methodological approach guidelines to all the design decisions involved in the creating of a text classification system. While an overall structure of a text classification system is well defined the information on how to select a feature extraction, feature selection and classification engine algorithms based on the task at hand is scarce and is not systematically united in a methodology of text classification system design. For the most part the design of a text classification system is not a science, but more of an art with the results being evaluated by extensive experimentation. Therefore there is a missing link between gathering data for the training set, establishing the goal of the classification and specific recommendations on

the optimum choices of a feature reduction algorithm, classification engine and their respective settings. There are some very specific suggestions available for some very specific cases, such as the case of high bias data. However a more general and systematic methodological guidance would be highly useful.

Also, as it has been noted above, if the amount of data on a particular topic is overwhelming and also bad data (intentionally distorted data, spam or just low quality material) exist in plentiful supply then classification of texts by quality is a way to further filter down the amount of information for users of a text classification technology. There are problems in this research domain, such as subjective definitions of quality and noise in the training data that is related to such subjectivity, however much benefit can be extracted from creation of a high performance text quality evaluation system in multiple industrial applications

4. Feature extraction

When approaching the field of automatic text classification the first problem is that any mathematical methods that can be used for the task of classification only operate on numbers and not on long, unstructured passages of text. Therefore, before any mathematical operations can be done on the text, some algorithms must be used to extract (Fig. 2) some kind of numerical information (features) of the classifiable text that are relevant to the classification (Sebastiani, 2002).

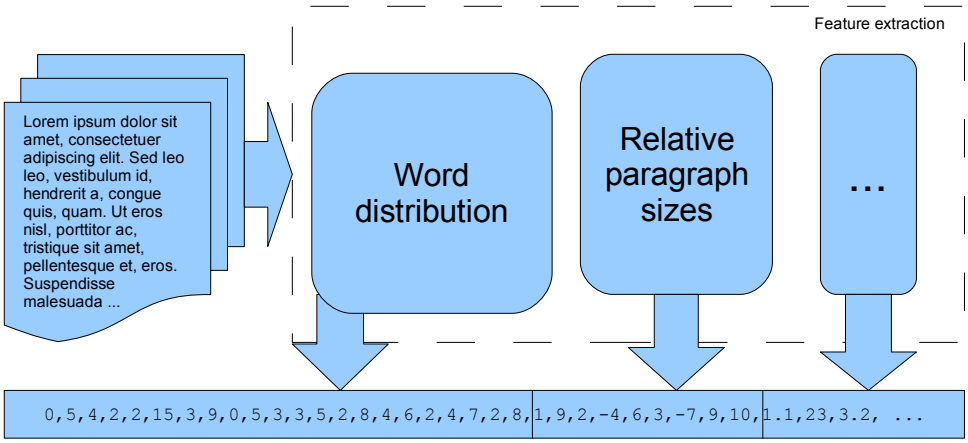

Figure 2: Feature extraction

The most common feature used is word frequency distribution. In preparation for the feature extraction the algorithm would look at all available texts (usually limited to the set of training documents) and create a dictionary of all words that occur in these texts. Each word in the dictionary is then allocated a place in the output feature vector (a dimension). When features of a document are extracted, this value in the feature vector will represent the number of times this word has occurred in the document. Minor modification of this method involves only putting "1" in the vector if the particular word is in the document and "0" if it is not, or dividing the number of times the word has occurred in this document by the average number of times it has occurred over the whole collection of documents (Haykin, 1998). Other methods (Wibowo & Williams, 2002) recommend to only take in account word distribution in the beginning of the document.

A relatively new development in generic text classification is the use of character or word sequence frequencies (Cancedda, Gaussier, Goutte & Renders, 2003) instead of the word frequencies. In character sequence distribution the frequency of a particular combination of characters is tracked through the texts. Word sequence distributions bring that to another level by tracking sequences of word stems. With this method occurrences of phrases “main body of text classification” and “main text body” would add no similarity to the texts as the word sequences are completely different (“main-body” + “body-text” + “text-classification” versus “main-text” + “text-body”).

If we can detect any kind of structure in the text being classified (for example HTML structure), features of that structure can be extracted and fed into the classification algorithm (Calado *et al.*, 2003; Yi & Sundaresan, 2000)

Additional features can be extracted from the classifiable text, however nature of such features should be highly dependent on the nature of classification to be carried out. If web sites need to be separated into spam and non-spam websites, then the word frequency distribution or the ontology is of little use for the classification, because of widespread tactics by the spammers to copy and paste mixture of texts from legitimate

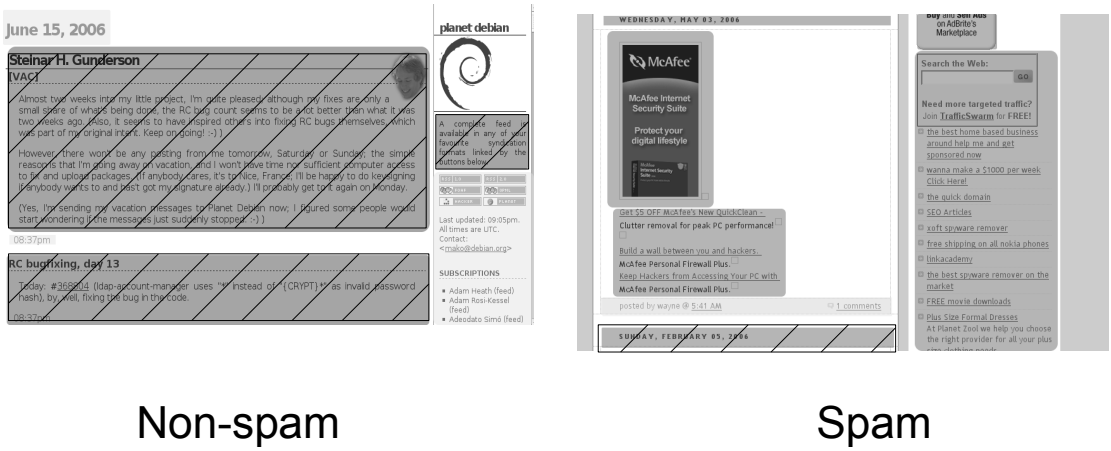

Figure 3: Screenshots of regular and spammy web site (area-coded)

web sites in creation of their spam web sites (Fetterly, Manasse & Najork, 2005).

In situations such as this, it is advisable to consult with the experts of such classification and extract some of their knowledge for use in the feature extraction phase. In the web spam example it is possible to detect advertisements on the page and compute how many pixels on a computer screen would be used by advertisements versus the content of the page when such page would be displayed normally in a browser. As can be seen above (Fig. 3) it is often possible to distinguish content oriented sites (likely to be non-spam) and advertisement oriented sites (likely to be spam) by just looking at the proportions of the content (dark, hatched), navigation (light) and advertisement (dark, clear) on the first screens of the page. Navigation and advertisement blocks can be detected (Shih & Karger, 2004; Song, Liu, Wen & Ma, 2004) using linking structure. Other web related features can be extracted using special wrapper programs (Laender, Ribeiro-Neto, da Silva & Teixeira, 2002; Meng, Hu & Li,

2003) that mine web pages for specific data bits and are very useful in semi-controlled subsets of the web, such as multiuser blogging sites. For regular hypertext, other linking features (Furnkranz, 1999) can be extracted, but the gains can be low and unstable because the Web is very noisy.

While creating text classification applications one must keep in mind that the classification algorithm does not actually read the text – it only looks at the features that are extracted for the algorithm from the text. If a human expert can not classify a document only from its feature vector and its explanation, then it is probable that the classification algorithm will not be able to do that as well.

5. Natural language processing

Natural language processing approaches can be applied both to feature extraction and feature reduction phases of the text classification process. Linguistic features can be extracted from texts and used as part of their feature vectors (Hunnisett & Teahan, 2004). For example (Stamatatos, Kokkinakis & Fakotakis, 2000) parts of the text that are written in direct speech, use of different types of declinations, length of sentences, proportions of different parts of speech in sentences (such as noun phrases, preposition phrases or verb phrases) can all be detected and used as a feature vector or in addition to word frequency feature vector.

Natural language processing can also be used in ways that encompass both feature extraction and reduction, for example, tools can be used to identify keywords (Zhang, Zincir-Heywood & Milios, 2005c) from a text document or even create a semi-structured summary of the text. It can be shown (Ma, Shepherd & Nguyen, 2003) that feature extraction from such condensed forms of the original documents can reduce the dimensionality of the input vector without reducing the classification performance.

The word sequence feature extraction that was described above and also most of feature reduction methods described in the next section use knowledge from the field of linguistics and natural language processing. However still more cooperation between linguistics and text classification scientists could possibly bring news ways of incorporating linguistic knowledge into the design of feature extraction, feature reduction and even classifier parts of a text classification engine.

6. Feature reduction

In a typical text classification approach for the classification of text articles in English, the word frequency is used as the primary part of the feature vector. This typically produces feature vectors with dimensionality in the order of tens of thousands of dimensions (Sebastiani, 2002). The computational complexity of any operations with such feature vectors will be proportional to the size of the feature vector (Yang & Pedersen, 1997), so any methods that reduce the size of the feature vector while not significantly impacting the classification performance are very welcome in any practical application. Additionally, it has been shown that some specific words in specific languages only add noise to the data and removing them from the feature vector actually improves classification performance (Yang & Pedersen, 1997).

The set (Liu & Motoda, 1998; Sebastiani, 2002) of feature reduction operations involves a combination of three general approaches:

1. Stop words;

2. Stemming;
3. Statistical filtering.

In any language there are many words that convey little or no meaning, but are required by the grammar structure of the language; these words are called “stop words”. As an example in the English language words like: “a”, “the”, “but” and many others are considered to be such stop words. It is common practice to exclude stop words from the feature vector. Stop word lists can be used or the stop word can be determined from their frequency, which is said to be more efficient and language independent (Ho, 1999; Wilbur & Sirotkin, 1992).

Second way of traditional feature reduction is the use of stemming to reduce frequencies of words with a common root to a single feature, for example, if the document would contain 7 instances of the word “house”, 3 instances of the word “houses” and 2 instances of the word “housing”, then after stemming reduction these three separate features would be reduced to only one that would describe that words with the root similar to “house” occurred in the document 12 times (7+3+2). Traditionally Porter's algorithm (Hull, 1996) is being used for stemming in English and other algorithms (Braschler & Ripplinger, 2004; Xu & Croft, 1998) are devised for other languages.

Statistical filtering practices are used to select those words that have higher statistical significance. Many different statistical methods are being researched and used for feature vector filtering, but the main difference between these methods is how much information about the source data is being used. It is possible to calculate generic statistical significance of a word in relation to how different its use frequency is in different documents, but more sophisticated algorithms also take into account the proposed classification of said documents and are essentially computing statistical significance of words in specific categories.

Most represented (Sebastiani, 2002) statistical filtering approaches are: odds ratio, mutual information, cross entropy, information gain, weight of evidence, χ^2 test, correlation coefficient (Ng, Goh & Low, 1997), conditional mutual information maximin (Wang & Lochovsky, 2004), and conformity/uniformity criteria (Chen, Lee & Hwang, 2005). Yang (Yang & Pedersen, 1997) and Mladenic (Mladenic & Grobelnik, 2003) compared some of those methods. In simple terms, most formulas give high scores to words that appear frequently within a category and less frequently outside of a category (conformity) or to the opposite (non-conformity). And additionally higher scores are given to words that appear in most documents of a particular category (uniformity).

Another way of reducing the feature vector is through the use of genetic computing (Zhang *et al.*, 2005a). However, in most applications the use of genetic computing will use up more resources then the resulting feature reduction could spare during the production run of the system.

After a good application of feature reduction algorithms one can expect to bring the size of a typical feature vector down from hundreds of thousands of dimensions to a few thousands of dimensions (Chen *et al.*, 2005). However some research (Riloff, 1995) shows that during feature reduction some subtle information is lost that could be useful for enhancing the precision of classification. So feature reduction in most cases is a speed versus precision tradeoff.

7. Statistical classification

The most widely used type of classifier is Naïve Bayesian classifier. Among others statistical classifiers (Zhang, Zhu & Yao, 2004) Bayesian classifier is the simplest, but it is still very effective and the due to its simplicity is also the single most researched classifier – it appeared in almost all articles that the author has read on the text classification related topics. Naïve Bayesian classifier assumes (naively) that features of the input feature vector (usually word distribution) are statistically independent (Rish, 2001; Sebastiani, 2002). In a basic form for two possible classes (for example, spam and non-spam texts), the Naïve Bayesian probability of a text being spam is equal to probability of finding its feature vector components in a spam text multiplied by the probability of any text being a spam text (i.e. the ratio of spam texts in the collection) divided by the general probability of a particular feature vector component occurring in a text. During training, features of each document of the training set are recorded directly – if the text is know to be spam, then its features are added both to spam text feature probabilities and to general text feature probabilities. Features of non-spam texts are added only to general text feature probabilities.

The probability of any text being a spam text is an estimated parameter of the algorithm – the larger this probability is the larger will be the number of texts classified as spam. Increasing this number will decrease the number of false negatives (i.e. spam texts classified as non-spam), but will also increase the number of false positives (i.e. non-spam texts classified as spam).

The structure of the Naïve Bayesian classifier makes it easy to encode some expert knowledge into the learning data set – for example, if experts agree that the word "Viagra" appears in spam much more frequently then in non-spam texts, then we can increase the spamminess probability of this word directly.

Large body of research (Nigam, McCallum, Thrun & Mitchell, 2000; Zheng & Webb, 2000) in text classification involves improving on the original Naïve Bayesian classification. Naïve Bayesian is also the canonical classifier (Rish, 2001) – every new text classification approach is tested against it first (Pant & Srinivasan, 2005).

8. Functional classification

If we think of every number in the feature vector as a coordinate in a dimension, then every document can be represented as a dot in a multidimensional space where the number of dimensions is equal to the number of features in the feature vector. Such interpretation allows to use geometrical ways of classification that can also be more easily presented visually.

One of the simplest geometrical (or functional) classifiers is the k Nearest Neighbours (kNN) classifier (Kwon & Lee, 2003). The idea of this classifier is very simple – in the multidimensional space we find the dot that represents the document being classified and look around to find out what other dots are nearby. Only k number of nearest neighbours are considered. If they all belong to the same category then the new document also will be categorised to that category. Otherwise the distribution of categories of the nearest neighbours determines the probability of the document belonging to the category. In other words, if, out of the 5 nearest neighbours, 4 belong to class A and 1 belongs to class B, then the new document is classified to class A with 80% certainty. Modifications of this algorithm also take in account the distance to the

neighbours in determining the certainty and/or the category. While the classifier is very simple, it is surprisingly effective and has also received a lot of research (Sebastiani, 2002).

Currently one of the most actively researched classifiers are the Support Vector Machines (SVM) (Tresch & Luniewski, 1995; Zhang, Chen & Lee, 2005b). If we visualise two classification classes as two blobs of points and imagine a border area between the classification classes, then we can identify documents that are the boundary examples of each class. If we find a vector that goes through the points in space that represent those boundary documents so that all documents of the category are on one side of this vector, then this will be a support vector. Mathematically, by averaging two support vectors from two categories, we can determine a vector that will lie roughly in the middle between the categories and that can be used to categorise new documents based on that on which side of this vector the new document is.

A number of different methods exist for determining a good support vector for a category and for calculating to which class a document belongs to from a set of support vectors for multiple classes in multidimensional space, selecting the optimal kernel and setting appropriate parameters for the kernels is one of the biggest challenges in use of SVM. SVM classification (Greevy & Smeaton, 2004; Yu, Han & Chang, 2002) receives the most attention in the text classification field among the theoretical research.

9. Neural classification

Neural network is a massively parallel distributed processor made up of simple processing units (neurons), which have a way for storing knowledge from experience and making it available for use. Knowledge is acquired by the network from its environment via a learning process and stored in interneuron connections (synaptic weights) (Haykin, 1998).

In principle use of the neural network for any classification task is straightforward: a neural network is taken, data of the feature vector is fed to the inputs of the network and categorisation comes from the outputs. Each output is directly assigned a category – if the strongest signal comes out of the neural network on the output number 3, for example, then the object being classified belongs to the third category. The difference in strength between the strongest output signal and other output signals indicate the confidence the network has in this classification (Sebastiani, 2002). If no output is strong enough, then the classification can be rejected (Fumera, Pillai & Roli, 2003) to improve reliability of the result.

However the fundamental problem in the use of a neural network is making the actual design of the network. Theoretically it is possible to construct neural networks of any complexity, but it is very hard to mathematically predict if a given neural network design will be able to excel in a particular classification task. Given this complexity the researchers have concentrated on simple and predictable neural network designs for most practical tasks in the field of text classification and only use more complex designs in the newer and more complex fields of image and speech recognition.

Standard models are used when there is little knowledge about the nature of the problem, but sometimes there is some expert knowledge about the documents being classified or about the structure of the classes that can be incorporated into the design of the classifier. For example, if we know that we need to classify a huge number of text

articles into a hierarchical topic structure, then the structure of the classifier can be adapted to that.

Research shows that for hierarchical topic classification a system of hierarchical perceptrons (Chen *et al.*, 2005) achieves better performance (higher F measure) then a single (larger) perceptron or a Bayesian approach. In a hierarchical classifier the document is first classified by a top level classifier which classifies the document into one of the top level categories. After that the document is passed down to the next classifier that is specific for this category. In such a way the document travels down the classifier hierarchy until it reaches the bottom layer where its final category is determined. This structure works well because each of the classifiers only needs to know how to classify documents within its narrow field of knowledge – its category (Dumais & Chen, 2000; Ruiz & Srinivasan, 2002). Less processing power is used, because only one branch of classifiers is activated at a time. Additionally having separate classifiers at different levels allows to reduce features even further by allowing different features to be significant for categories, for example in the sports category words such as "puck", "quarterback" or "offside" could be very valuable to determine the specific type of sport the article is about, but more generic sport terms such as "training" or "ball" could be less useful.

10. Learning and evaluation

A classifier by itself has no knowledge. Any knowledge that is required for classification must come to a classifier either by directly translating expert knowledge or from learning. Two major types of learning are supervised learning and unsupervised learning.

How many examples do we need and how long do we need to train the network with until it is good enough and what exactly "good enough" means in this sense? First of all, it is definite that the more examples we can get for the network to learn from, the better the results will be, just like with typical learning – practice makes the master. If we can provide all the possible input vectors and correct classifications for them, then that is the best possible scenario, but in that case much more effective ways of storing and recalling that information are available (memory based classification) and there is little or no need to use more complex classifiers. Where text classification shines is the ability of generalisation – a way to provide a classification result for an input that the classifier has not seen in training. To measure success in these areas, two basic measures are used: precision and recall (Haykin, 1998). First a confusion matrix is computed. For a simple case of two categories it is a 2x2 matrix where test cases are distributed as follows: first cell is the number of test cases that were correctly assigned to the first category (True Positive), the second is the number of test cases that should have been in the first category, but were classified as belonging to the second one (False Negative), and third and forth cell respectively for category one that should have been two and for correct category two test cases.

True Positive	False Negative
False Positive	True Negative

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Figure 4: Confusion matrix, precision and recall

The formulae for precision and recall are quite simplistic and more complex measures are often used, such a F_1 measure and Maximal Figure-of-Merit (Gao, Wu, Lee & Chua, 2003).

In the simplest supervised learning case all the example data (pairs of input vectors and correct output vectors) is randomly divided into three parts – training, testing and validation data sets. Then the training starts – the network is shown examples from the training set in random order and any errors are corrected by backpropagation. The process goes through all the training set data multiple times until the recall on the last iteration is higher than a predetermined minimum recall threshold. Sometimes a fixed number of iterations is used.

When the training is deemed to be finished, the testing process starts. In the testing process all examples from the testing data set are given to the classifier and the precision over the testing data is computed. If this precision is lower than a predetermined minimum generalisation threshold (usually around 70-80%), then the system goes back to the learning stage (usually for a fixed number of iterations).

The final stage is the verification stage. The process is similar to the testing stage – average error across the whole data set is computed. What is different about the verification stage is its outcome. The precision measure (often, the F_1 measure) computed at the verification stage is the final precision of the classifier and is the canonical figure by which different classification systems can be compared. If the value is good then the network is ready to be used and the learning is complete. However if the verification precision of the network is unsatisfactory then no amount of learning will help it – to improve the result the structure of the classifier system must be changed.

Alternative ways to learning involve more effective use of the example data. This is important because even manually classifying a thousand examples (a low number for neural network training) is a very time consuming task. Also, only a qualified human expert can do this task, which can induce a heavy monetary cost on the procedure. Therefore getting maximum impact from relatively small number of examples is a critical problem. One way of doing this is to train several classifiers at the same time with different splits of the same training data and selecting the one with the best verification results.

In an unsupervised learning scenario there is no teacher and thus no direct feedback on the actions of a classifier. Instead the classifier tries to make a good representation of the input vector in the output and a task-independent measure is used to determine the quality of such representation. For example in Web clustering (Adami, Avesani & Sona, 2003) if we have a large number of documents, but we do not strictly know the full classification structure we want to classify them into, unsupervised learning can be

used. What unsupervised learning would do is simply grouping together web pages with similar content or topic. The quality measure could be the confidence of classification given a fixed number of categories to use. However, unsupervised neural networks have been shown (Zhang, 2007) to extract patterns where there are none, so care must be given to problems of validation to avoid over-fitting.

Unsupervised learning can be combined with supervised learning (Chuan, Xianliang, Mengshu & Xu, 2005) for feature reduction purposes before the classification by a classical back-propagation neural network or to compensate the for lack of negative classification examples in a two-class classification scenario (Yu *et al.*, 2002). Semi-supervised learning combines known classifications and unknown classifications to expand on the range of possible classes (Carsten Lanquillon, 2000; Nigam *et al.*, 2000).

Another way to make most of existing labelled examples is to use the boosting approach. The idea is to train multiple classifiers sequentially with each next classifier focussing on the examples that previous classifiers performed badly on. A committee of classifiers is formed this way with different classifiers supporting each other. AdaBoost (Schapire, Singer & Singhal, 1998) is a popular example of this approach.

11. Conclusions

Text classification is an mature area of research that has been revitalised in 1980s by the increase of information flow available. It has seen large attention especially due to the high growth rate of Internet and the importance of Internet search engines and generic classification of content on the Web. Process of text classification is well researched, but still many improvements can be made both to the feature preparation and to the classification engine itself to optimise the classification performance for a specific application. Research describing what adjustments should be made in specific situations is common, but a more generic framework is lacking. Effects of specific adjustments are also not well researched outside the original area of application. Due to these reasons, design of text classification systems is still more of an art than exact science.

References

Adami, G., Avesani, P. & Sona, D. (2003). Clustering documents in a web directory. In *Proceedings of the 5th ACM international workshop on Web information and data management* (pp. pp. 66-73). : ACM Press, New Orleans, Louisiana, USA

Braschler, M. & Ripplinger, B. (2004). How Effective is Stemming and Decompounding for German Text Retrieval?. *Information Retrieval*, *7*, pp. 291-316.

Calado, P., Cristo, M., Moura, E., Ziviani, N., Ribeiro-Neto, B. & Goncalves, M.A. (2003). Combining link-based and content-based methods for web document classification. In *Proceedings of the twelfth international conference on Information and knowledge management* (pp. pp. 394-401). : ACM Press, New Orleans, LA, USA

Cancedda, N., Gaussier, E., Goutte, C. & Renders, J.M. (2003). Word sequence kernels. *Journal of Machine Learning Research*, *3*, pp. 1059-1082.

Carsten Lanquillon (2000). Learning from Labeled and Unlabeled Documents: A Comparative Study on Semi-Supervised Text Classification. In *Proceedings of PKDD-00 4th European Conference on Principles of Data Mining and Knowledge Discovery* (pp. p. 490--497). : Springer Verlag Heidelberg DE

Chen, C., Lee, H. & Hwang, C. (2005). A Hierarchical Neural Network Document Classifier with Linguistic Feature Selection. *Applied Intelligence*, *23*, pp. 277-294.

Chuan, Z., Xianliang, L., Mengshu, H. & Xu, Z. (2005). A LVQ-based neural network anti-spam email approach. *Operating Systems Review (ACM)*, *39*, pp. 34-39.

Dumais, S. & Chen, H. (2000). Hierarchical classification of Web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. pp. 256-263). : ACM Press, Athens, Greece

Evgeniy Gabrilovich, F. S. (2006). *Text categorization bibliography*. Retrieved from http://www.cs.technion.ac.il/~gabr/resources/atc/atcbib.html.

Fetterly, D., Manasse, M. & Najork, M. (2005). Detecting phrase-level duplication on the world wide web. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. pp. 170-177). : ACM Press, Salvador, Brazil

Fumera, G., Pillai, I. & Roli, F. (2003). Classification with Reject Option in Text Categorisation Systems. In *Proceedings of the 12th International Conference on Image Analysis and Processing* (pp. pp. 582-587). : IEEE Computer Society

Furnkranz, J. (1999). Exploiting Structural Information for Text Classification on the WWW. In *Proceedings of IDA-99 3rd Symposium on Intelligent Data Analysis* (pp. p. 487--497). : Springer Verlag Heidelberg DE

Gao, S., Wu, W., Lee, C. & Chua, T. (2003). A maximal figure-of-merit learning approach to text categorization. In *Proceedings of SIGIR-03 26th ACM International Conference on Research and Development in Information Retrieval* (pp. pp. 174-181). : ACM Press New York US

Greevy, E. & Smeaton, A.F. (2004). Classifying racist texts using a support vector machine. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. pp. 468-469). : ACM Press, Sheffield, United Kingdom

Haykin, S. (1998). Neural networks: a comprehensive foundation. : Prentice Hall PT.

Ho, T. K. (1999). Fast Identification of Stop Words for Font Learning and Keyword Spotting. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition* (pp. pp. 333-336). : IEEE Computer Society

Hull, D. A. (1996). Stemming algorithms: a case study for detailed evaluation. *Journal of the American Society for Information Science*, *47*, pp. 70-84.

Hunnisett, D. S. & Teahan, W.J. (2004). Context-based methods for text categorisation. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. pp. 578-579). : ACM Press, Sheffield, United Kingdom

Kwon, O. & Lee, J. (2003). Text categorization based on k-nearest neighbor approach for Web site classification. *Information Processing and Management*, *39*, pp. 25-44.

Laender, A. H. F., Ribeiro-Neto, B. A., da Silva, A. S. & Teixeira, J.S. (2002). A brief survey of web data extraction tools. *SIGMOD Rec*, *31*, pp. 84-93.

Liu, H. & Motoda, H. (1998). Feature Selection for Knowledge Discovery and Data Mining. : Kluwer Academic Publisher.

Ma, L., Shepherd, J. & Nguyen, A. (2003). Document classification via structure synopses. In *Proceedings of the Fourteenth Australasian database conference on Database technologies 2003 - Volume 17* (pp. pp. 59-65). : Australian Computer Society, Inc, Adelaide, Australia

Meng, X., Hu, D. & Li, C. (2003). Schema-guided wrapper maintenance for web-data extraction. In *Proceedings of the 5th ACM international workshop on Web information and data management* (pp. pp. 1-8). : ACM Press, New Orleans, Louisiana, USA

Mladenic, D. & Grobelnik, M. (2003). Feature selection on hierarchy of web documents. *Decision Support Systems*, *35*, pp. 45-87.

Ng, H. T., Goh, W. B. & Low, K.L. (1997). Feature selection, perception learning, and a usability case study for text categorization. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. pp. 67-73). : ACM Press, Philadelphia, Pennsylvania, United States

Nigam, K., McCallum, A. K., Thrun, S. & Mitchell, T. (2000). Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, *39*, pp. 103-134.

Pant, G. & Srinivasan, P. (2005). Learning to crawl: Comparing classification schemes. *ACM Transactions on Information Systems*, *23*, pp. 430-462.

Riloff, E. (1995). Little words can make a big difference for text classification. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. pp. 130-136). : ACM Press, Seattle, Washington, United States

Rish, I. (2001). An empirical study of the naive Bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence* (pp. pp. 41-46). : T.J. Watson Research Centre, Seattle, Washington

Ruiz, M. E. & Srinivasan, P. (2002). Hierarchical Text Categorization Using Neural Networks. *Information Retrieval*, *5*, pp. 87-118.

Schapire, R. E., Singer, Y. & Singhal, A. (1998). Boosting and Rocchio applied to text filtering. In *Proceedings of SIGIR-98 21st ACM International Conference on Research and Development in Information Retrieval* (pp. p. 215--223). : ACM Press New York US

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*, pp. 1-47.

Shih, L. K. & Karger, D.R. (2004). Using urls and table layout for web classification tasks. In *Proceedings of the 13th international conference on World Wide Web* (pp. pp. 193-202). : ACM Press, New York, NY, USA

Song, R., Liu, H., Wen, J. & Ma, W. (2004). Learning important models for web page blocks based on layout and content analysis. *ACM SIGKDD Explorations Newsletter*, *6*, pp. 14-23.

Stamatatos, E., Kokkinakis, G. & Fakotakis, N. (2000). Automatic text categorization in terms of genre and author. *Computational Linguistics*, *26*, pp. 471-495.

Tresch, M. & Luniewski, A. (1995). Extensible classifier for semi-structured documents. In *Proceedings of the fourth international conference on Information and knowledge management* (pp. pp. 226-233). : ACM Press, Baltimore, Maryland, United States

Wang, G. & Lochovsky, F.H. (2004). Feature selection with conditional mutual information maximin in text categorization. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management* (pp. pp. 342-349). : ACM Press, Washington, D.C., USA

Wibowo, W. & Williams, H.E. (2002). Simple and accurate feature selection for hierarchical categorisation. In *Proceedings of the 2002 ACM symposium on Document engineering* (pp. pp. 111-118). : ACM Press, McLean, Virginia, USA

Wilbur, J. & Sirotkin, K. (1992). The automatic identification of stop words. *Journal of Information Science*, *18*, pp. 45-55.

Xu, J. & Croft, B. (1998). Corpus-based stemming using cooccurrence of word variants. *ACM Transactions on Information Systems*, *16*, pp. 61-81.

Yang, Y. & Pedersen, J.O. (1997). A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. pp. 412-420). : Morgan Kaufmann Publishers Inc, San Francisco, CA, USA

Yi, J. & Sundaresan, N. (2000). A classifier for semi-structured documents. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. pp. 340-344). : ACM Press, Boston, Massachusetts, United States

Yu, H., Han, J. & Chang, K.C. (2002). PEBL: positive example based learning for Web page classification using SVM. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. pp. 239-248). : ACM Press, Edmonton, Alberta, Canada

Zhang, B., Chen, Y., Fan, W., Fox, E. A., Goncalves, M., Cristo, M. & Calado, P. (2005a). Intelligent GP fusion from multiple sources for text classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management* (pp. pp. 477-484). : ACM Press, Bremen, Germany

Zhang, D., Chen, X. & Lee, W.S. (2005b). Text classification with kernels on the multinomial manifold. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. pp. 266-273). : ACM Press, Salvador, Brazil

Zhang, G. P. (2007). Avoiding Pitfalls in Neural Network Research. *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *37*, pp. 3-16.

Zhang, L., Zhu, J. & Yao, T. (2004). An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, *3*, pp. 243-269.

Zhang, Y., Zincir-Heywood, N. & Milios, E. (2005c). Narrative text classification for automatic key phrase extraction in web document corpora. In *Proceedings of the 7th annual ACM international workshop on Web information and data management* (pp. pp. 51-58). : ACM Press, Bremen, Germany

Zheng, Z. & Webb, G.I. (2000). Lazy Learning of Bayesian Rules. *Machine Learning*, *41*, pp. 53-84.

