A Model for Information Representation and Retrieval in Large Organizations

Arman Tajarobi

Carleton University Cognitive Science Technical Report
2010-03

cogscitechreports@carleton.ca
Institute of Cognitive Science
2201 Dunton Tower
Carleton University
1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6
Canada

INSTITUTE OF COGNITIVE SCIENCE

CARLETON UNIVERSITY

A Model for Information Representation and Retrieval in Large Organizations

CGSC6905 : METHODOLOGY ROTATION

August 2010

FINAL REPORT

Student: Arman Tajarobi

Methodology Rotation Supervisor: Jennifer Schellinck

Contents

1. Introduction

Since the 1970's, organizations have adopted a number of analytical tools that are commonly referred to as Business Intelligence. These tools have matured and work quite well with structured data, providing organizations with core analytics that derive intelligence from raw data. However, since the 1990's, organizations have seen an exponential growth in their unstructured information: PowerPoint presentations, Word documents, PDFs, emails, blogs, etc. Various analyst groups report that unstructured data doubles every three months in a typical organization today. Employees require consistent and predictable access to this growing knowledge to effectively do their jobs. However, as each new piece of content is added, their ability to find the information they need diminishes. Information derived from analysis is a key component for day to day activities. Several important factors are at play here. Deriving competitive advantage requires that analytical tools and results be accessible to a larger proportion of the workforce for a broader range of analyses. To allow more people use these tools, they have to be built on a familiar and intuitive framework. Increasing competition, customer demand and pricing pressure are forcing businesses to accelerate processes and eliminate delays. The foundation of text analytics is to assist the human brain in the analysis of information in order to speed up the retrieval of relevant information and convert it to intelligence. Moreover, as more and more tools become available to consumers to share their knowledge, opinions and sentiments on the Internet, organizations realize that there is business value in this information. Technologies to support the analysis of Consumer Generated Media (CGM) in near real-time are rapidly becoming key competitive tools. For organizations operating on a global basis, this analysis must integrate technologies to perform the analysis on multilingual content. New legislation is also having a dramatic impact on organizations. New legal requirements are forcing organizations to accumulate more and more unstructured content such as emails, reports, memos, conversations with customers, Customer Relationship Management (CRM) notes, etc. and to use more intelligent search tools to retrieve the most current and detailed information on operations.

In the evolution of information management, organizing or categorizing information into an intuitive topical hierarchy or taxonomy has proven to be an efficient and productive way for end users to not only find, but also to discover the information crucial to enterprise productivity.

In this report, the focus will be on the generation and use of taxonomies for knowledge representation and discovery in organizations. A model for the taxonomic organization, and subsequent use and manipulations, of compound terms is presented. More specifically, the presented model envisions the organization of compound terms into multiple taxonomies which can be dynamically related in various ways based on the context of usage. We will be referring to the set of interrelated taxonomies as *facetted taxonomies*, following the traditional usage of this term, with the nuance that the set is not static and is dynamically adapting to the usage context.

2. Automated Document Classification: An Overview of the State of the Art

Taxonomies are inherently related to information organization, or classification. Within automated and semi-automated contexts, automatic classification of content based on predefined taxonomies is an important component of information management. Automated text classification is an important area of applied research in the fields of computational linguistics, as well as computer science, artificial intelligence, and areas involving text mining, machine learning, and intelligent agents. Document classification is an area that has received a lot of attention in the past few decades, evolving into a mature and practical domain. We will begin the review of the state of art by a brief look at the manual classification, which predates the automated processes by several decades.

Manual text classification is an old area of interest, mainly developed within the Information and Library Sciences, which usually refer to the task as “cataloguing” (or “categorization”). Cataloguing in Library Sciences refers to the process of organizing information with the aim of facilitating access to that information (Chan, 2000). Cataloguing is assumed to have existed in one form or another from the time of first libraries, which date back to at least 1200 B.C. (Svenonius, 2000).

Since the task of manual document classification can be quite complex and cumbersome, until recently the application of the process has been reserved for information structures of high value, such as books, scientific articles, official documents, and such. However, with the advent of automated and intelligent systems, and the exponential growth of all kinds of information structures in textual format, the process is now used in many other places.

This overview focuses on automated document classification (ADC). The aim is to lay out the main approaches used for ADC, with particular emphasis on machine learning approaches. It will outline the advantages and disadvantages of the main approaches and, where data is available, will compare them to each other.

The paper is organised in the following manner:

- The first few pages will give a high level definition of document classification, including some examples that include practical and industrial implications with the aim of putting into perspective some of the challenges of ATC, as well as the kind of criteria that should be used to evaluate ATC systems.
- The next section will include some background information on rule-based and lexicon-based approaches to document classification. This background will complement the previous section and allow for formalising a set of evaluation criteria for this domain.
- The following section explores the most common machine learning approaches in document classification. A high level description of each is provided, outlining the weaknesses and strengths of each approach, and, where available, some evaluation data.
- The last section will include a synthesis of all the machine learning approaches and a summary discussion of this section.

2.1 Document Classification

Document classification, also commonly referred to as document categorization[1] (as well as "indexing"), is the process of organizing documents into predetermined classification schema. As with other classification systems, document classification systems group *similar* documents together with the aim of creating a structure that will make managing of large sets of documents easier.

Similarity can be a vague notion unless its parameters are known. For instance, a rudimentary document classification system could be based on the size of documents (i.e. large, medium, small). In this case, all the documents classed as *large* are said to be similar in size. Similarity could also be based on other criteria such as language of documents, their origin, their date of

[1] Categorisation is often used interchangeably with classification in the context of ADC. Technically, the term classification differs from categorisation. Classification is the assignment of a document to a class. Categorization is the assignment of a class (or category) to a document. The difference is subtle, but in practice, especially when dealing with paper material, a single document cannot be classed into more than one spot (class). Therefore classification is restricted to one class per document, where as there is no upper limit to the number of categories that can be assigned to a single document. However, with electronic documents, since the assignments are virtual in both directions, the two operations can be seen to result in the same thing. In this paper, I will use the two terms to mean the same thing, unless otherwise indicated. A third term that is also used synonymously with classification and categorisation is "indexing". Although back-of-the-book indexing is usually well distinguished, some document types such technical documents may contain terms that define their "aboutness". This characteristic has been exploited by the simple approach of searching for specialised terms in documents and assigning the terms in the same way as one would assign categories or class tags to documents. Therefore, indexing sometimes is used to mean categorisation, which itself is often used to mean classification!

creation, and so on. However, within the Information and Library Sciences, document classification mainly refers to the classification of documents according to their subject matter or topics. In other words, *what is the document about*?

Classification based on *aboutness* requires a set of skills that can be highly specialised. For instance, librarians in charge of cataloguing books and other library material need to go through several years of training. Even then, their task would be almost impossible to perform if publishers did not provide a good amount of classification information with published materials. In the case of libraries, usually the cataloguing systems of the Library of Congress (LOC) or the Dewey Decimal Classification (DDC) system are used (Chan, 1992). Libraries have been very slow in adopting automatic classification systems, partly because of the physical nature of their documents (mostly paper), and also because, as will be discussed in the following paragraphs, the volumes they need to deal with are relatively small.

Another big sector where document classification is part of the everyday process is the publishing industry. These companies usually have teams of "indexers" or "cataloguers". I worked for a company that helped organizations such as Proquest and the American Psychology Association (APA) to automate a part of their cataloguing process[2]. For example, at APA a team of highly trained indexers (many held doctoral degrees in psychology) was in charge of cataloguing the vast volumes of publications in psychology that go through APA each day. A high degree of knowledge of the subject domain is required to perform such a task. This is what makes the task of document classification highly complex and costly. For instance, at APA, the average cost of processing a single document was estimated to be over $100 (USD). They processed several hundred documents each week, which is actually rather modest compared to some other publishing compnies who deal with several thousand documents each day. Beginning in the mid 1990's, most of the large and medium sized publishing organisations started to automate at least a part of their classification processes.

More recently, there has also been great interest in what is commonly referred to as Consumer-Generated Content (CGC) which is produced at a dizzying rate on the Internet. These include

[2] See www.nstein.com

personal web pages, blogs, comments, feedbacks, opinions, personal profiles, and all the other publicly available data on the Internet that is produced by the general public. The information contained in these documents is usually of little value on a document-by-document basis, but together they contain valuable information about trends, tendencies, and the all-important *public opinion.* Traditional methods of mining public opinion are quickly changing. Polling companies are finding that instead of calling people on the phone (or stopping them in shopping malls) to ask dozens of questions about a product or event, they can simply tap into online resources, where people are freely volunteering their opinions about products, politicians, and other subjects of interest. There are many different approaches to mining public opinion on the web, but classification is usually an important part of most approaches. Of course, the sheer volume makes any manual classification unthinkable. As for automatic classification, factors such as lack of consistency and reliable structure have made this domain extremely challenging.

A last domain of particular interest here includes what may be called private and semi-private documents, which have also seen a great surge in volume following the increasing use of digital data within private and public organizations and at home. Semi-private documents include documents inside a company, organisation or institution and reside within institutional networks (Intranets) where access is restricted to only designated individuals. There are virtually hundreds of institutional document types, including e-mails, forms, memos, business plans, financial statements, etc. that are used to conduct and support daily operations within organisations. These exist in extremely high volumes, and managing them in a timely manner is usually of critical importance.

Private documents include personal documents in the possession of individuals. E-mail is a good example, but many other document types can be included here. Basically any document on someone’s personal or work computer is considered private data. Usually each person has a personalised method of classifying their documents, and today most people are finding it impossible to manually classify their own documents in a comprehensive and consistent manner. While less critical than semi-private documents, most everyone recognizes that facilitating the organisation of personal documents can contribute significantly to efficacy. There is certainly a huge pool of users awaiting the automated systems to be mature enough to be implemented

within this context. This can be considered a domain for the future where aspects such as high customizability and flexible organizational schemes will be at the forefront.

Regarding organizational schemes, in document classification systems, documents are typically arranged in a hierarchical tree structure called a classification schema or a taxonomy. A taxonomy is a hierarchical classification scheme of information components (for example, terms, concepts, entities) and their interrelationships. Taxonomies support information management requirements by enabling end users to discover and access information and allowing content managers to organize their information assets.

Taxonomies come in all sizes and forms. Some are extremely simple, containing as few as only two nodes3, while others may be extremely complex, containing several thousand nodes, and facets to create multidimensional views4. Taxonomies are closely related to "ontologies," which, in turn, are key to the development and maintenance of taxonomies. According to the World Wide Web Consortium, an ontology defines terms, their properties and concepts (including relationships), which are used to describe bodies of knowledge. Accordingly, taxonomies are examples of ontologies; they provide the terms of a domain and represent a relationship among the terms (there also can be more than one relationship) as a structure. An ontology can be used to generate a taxonomy.

A taxonomy is often used as an inherent part of complex content management. It defines the organizing structure and, thus, guides the indexing, tagging or hyperlinking of content. Ideally, the taxonomy will be designed at the inception of a content management project. This is especially important if the heterogeneity of the content is significant, and if users will be challenged to access relevant content via normal (or even advanced) keyword searches.

Taxonomies provide alternative access paths to content that would otherwise be difficult to characterize as a question or keyword query. Taxonomies for document management usually are

[3] A sentiment analysis system only using Negative and Positive classes to classify documents is an example of a two node taxonomy.

[4] For example, the APA taxonomy mentioned earlier contains over 7,000 nodes, grouped in a tree structure of several levels. MeSH (Medical Subject Headings) is an example of a taxonomy with several facets (http://www.nlm.nih.gov/mesh/).

simpler than complex content management. Because document management often is focused within a business domain, or is a departmental application, this domain association can restrict the vocabulary. Also, documents in a document management system often are linked to a specific business transaction, customer account or project; therefore, the indexing and vocabulary usually are linked to the related business process. A search usually is faster and more effective because the vocabulary and business domains are narrower. Thus, taxonomies play a lesser discovery role in these document management. Complex content management and the associated complex taxonomy are inherent parts of a knowledge management program. The effort to build a taxonomy indicates the intention to establish the relevancy of the underlying content or documents to the end user. Taxonomies also can provide role-based relevancy of information, thus enhancing knowledge workers' decision making.

There is much more that could be discussed about taxonomies, but for the purpose of this paper, it suffices to mention that taxonomies are an integral part of classification and play an important role in any classification system.
This brief introduction to classification aimed at putting in context the rest of this section, which essentially deals with machine learning systems for text classification. The next section will start with the precursors of machine learning systems, notably rule-based and lexicon-based systems.

2.2 Towards Automation of Document Classification

Traditionally, within the field of automated document classification, documents were considered as a collection of words. Although this characterization has persisted until today to some degree, as the following sections demonstrate, approaches to dealing with documents have substantially changed.

The first attempts to automatically classify documents were essentially lexicon-based. The hypothesis was that any subject matter could be defined by a number of keywords that together would define that subject. For example, if some news articles are to be classed in the general classes of Politics, Sports, Entertainment, Economy, and Health, then lists of categorized words

(commonly referred to as “dictionaries”) containing words that are often used when talking about these subjects would suffice to class the documents with a relatively high degree of precision.

This simplistic approach is actually quite effective if the number of classes is small, and each class is distinct from the others, as in the example above. However, as soon as the number of categories increases and some of the *aboutness* of classes overlaps into other classes, then this approach fails to deliver good results.

This led to a series of refinements of the approach which included the use of the tf-idf (term frequency–inverse document frequency) measure (Salton et al., 1982). Tf-idf is essentially a measure of the importance of a word in a document. In its simplest form, tf-idf is calculated by dividing the number of times a word appears in a document by the number of total words in the document. So if the word “Politics” appears three times in a 100-word document, it will have a tf-idf of 0.03. The idea here is that words with a high tf-idf are probably too general to be used for classification, and words with a very low tf-idf do not have a high enough relative recurrence to be used for this purpose either. Usually a sort of upper and lower threshold is then set. This will enable the system to select words that are more significant and therefore improve on accuracy, while avoiding some of the problems such as overlapping of classes.
A further improvement to this technique was the introduction of tf-idf weights not only on a document, but on a collection of documents, or a corpus (Salton, 1991). Here, the frequency of a word in a document was compared to the frequency of that word in a given corpus. If the words appeared only in that document, they would have very high tf-idf for that document, making them excellent candidates for classification. However, if the words appear in all or almost all documents (stop words for example), then they are rejected as having no specific *aboutness* value.

Despite these improvements, these systems remained fairly inaccurate and difficult to manage. (see for example, Salton (For example, Salton 1991, Yang and Liu 1999), adding a new class would entail reviewing the dictionaries, and adding an additional entry to one class could create unpredictable results in other classes. However, the simplicity of these systems has allowed them to persist to this day. They remain the most intuitive approach to ADC.

Another variation of lexicon-based classifiers are rule-based systems (Norbert and Gerhard, 1984; Norbert et Al. 1991; Sasaki and Kita, 1998). Human-engineered rule-based classifiers require extensive developmental efforts, but have been successfully built to assign classification topics to documents. Rules significantly improve the flexibility and accuracy of ADC systems. Rules are simple to complex "If...Then" statements forcing the system to do something in specific situations. A rule has input parameters that trigger its activation. For example, a rule could state that if a particular word is seen in the first line of a document (i.e. its title), then the system must assign that document to a particular category.

Rule-based systems can achieve high degrees of accuracy in closed environments[5]. They also allow for a fair amount of flexibility and control over the system. The evolution of these systems can be planned in advance, and every action of the system can be easily traced back to a set of rules. The main drawback to rule-based systems is that they can only be used effectively in the environment for which they were built. Also, the creation and maintenance of rules is a very complicated task, making the systems highly dependent on domain experts. However, human-engineered rule-based systems remain the choice of most organisations that develop their systems in-house, and who are concerned about the questions of accuracy and control.

There's been several interesting projects aiming at the automation of the creation and maintenance of rules. Li and Yamanishi (2002) implemented a text classification system using stochastic decision lists. A stochastic decision list is an ordered sequence of IF-THEN-ELSE rules. The rules implement explicit knowledge about classification such as if a document contains certain words in the title then that document belongs to a particular class, else it belongs to another class. Such if-then-else rules can become quite sophisticated and powerful, employing vast quantities of formalized knowledge extracted from domain experts. The method has advantages of readability and refinability of acquired knowledge and automates the construction of decision lists on the basis of the principle of minimizing extended stochastic complexity, thus minimizing classification errors. The method achieves higher accuracy rates than human-

[5] A closed environment refers to an ADC environment where the taxonomy and document types remain fairly constant.

engineered rule-based systems but only on short documents. Chidanand et al (1994) describe use an “optimized rule-based induction” method for large document collections with the goal of discovering automatically classification patterns that can be used for general document categorization. While still in the category of rule-based systems, several machine learning techniques are used. I will cover machine learning in the future sections.

In summary, lexicon and rule-based systems can achieve high accuracy, but require important amounts of time and effort to get the initial system operational. Also, in terms of computing speed, these kinds of systems suffer as their databases grow: While executing a single rule or looking up a small number of dictionary items can be fairly quick, most mature systems contain gigantic databases, the management of which becomes an important challenge in itself. Lastly, these systems are designed with a “perfect” world in mind, where there are no typos, misspelled words, missing spaces between words, or other anomalies that, in reality, often appear in even very polished documents such as books and review publications. As one moves towards private, semi-private and Consumer-Generated Content mentioned in the last section, the quality of documents deteriorates significantly, making these systems less effective.

The rest of this paper deals with machine learning approaches in document classification. I will first discuss the criteria for evaluating classification systems and then give an overview of the most common machine learning approaches for document classification.

2.3 Evaluation Criteria

The preceding discussion demonstrates that classification systems are complex entities, whose evaluation has to encompass many different criteria. Unfortunately, the evaluation criterion most often used in academia is based on accuracy measures, neglecting other important factors. Although accuracy is important, it only exposes a part of the picture. An additional difficulty in comparing different systems arises from the fact that most of the evaluation data available are based on tests on a single system, or in some cases comparisons between two systems. As Yang et al. (1999) indicate, while there is a rich literature about individual methods, clear conclusions about cross-method comparisons are difficult to come by, because most often published reports

are not comparable. For instance, while both Joachim (1998) and Weiner et al. (1995) had evaluated the systems they were proposing, it is difficult to determine whether the performance of NNet by Wiener et al. (1995) is statistically better or worse than the performance of SVM by Joachims (1998) because different data collections were used in the evaluations of those methods, and no statistical significance analysis was conducted to verify the impact of the difference in data on the performance variation of these classifiers.

In addition, while a system may perform very well on one set of data, it may not do so well on another set. The reasons for this are varied and are related to the design and internal workings of each model. So basing evaluations on results announced for individual systems makes little sense.

Having said this, for this paper, there is no other choice but to limit the scope to data collected by previous researchers (the only other alternative being to run independent tests, which is clearly outside of the scope of this project). There are a few papers such as Yang's (1997, 1999) and Meyer et al. (2003) that provide some cross system analysis (all dealing exclusively with accuracy). This paper will synthesis these results and provide (as much as possible) some other criteria, based on personal experience, and what can be deduced from the available literature.
The following section defines each of these criteria, putting more emphasis on accuracy.

2.3.1 Accuracy

Following Rijsbergen (1979), accuracy is most often expressed as F-measure (F), which is calculated from precision (P) and recall (R), and expressed as:

F = 2(P*R)/(P+R)

Precision refers to the correctness of class assignments. Recall refers to the completeness of those assignments:

Recall = Correct classes found/total correct classes

Precision = Correct classes found / total classes found

For instance, if a document should be assigned to four classes (e.g. Politics, Elections, Health, and Environment) one would have a 100% F-measure if indeed the system assigned

it to those four classes, and ONLY to those four classes. However, if the system assigned the document to the classes *Politics*, *Environment*, and *Schools*, then Precision = 2/3 = 66% (system is correct for 2 of the 3 classes assigned), and recall = 2/4 = 50% (system found 2 correct classes out of the 4 possible). The F-measure, or accuracy, in this case is around 56% for this document.

There are several variations of this formula, but essentially F-measure expresses accuracy in terms of both recall and precision.

It should be noted that measuring accuracy is not an exact science. As Sebastiani (2002) confirms, "the subjectivity in the class concept definition makes analytical evaluation of text classifiers very difficult. Therefore empirical experiments have become the common text classification evaluation methods". The benchmarks are always based on human classification, which itself is somehow subjective and prone to errors. For instance, human classification variations between indexers are a well known fact, even if most indexing teams have very comprehensive guidelines and standards that everyone tries to follow. Another known phenomenon is that machine errors are much more visible than human errors (at least to humans). While a human may neglect to assign a document to a secondary class, or might assign it to a class that another human may not (but a third would!), machine errors are viewed as quite "unintelligent" by most humans.

2.3.2 Speed

If accuracy measures suffer from some subjectivity, speed, at least in theory, can be measured very accurately. However, in practice, there are many factors involved that make the task highly complicated. For one thing, we need to define what is meant by speed. Is it the amount of time it takes for an system to classify a document? Is so, where are the "start" and "end" points? Different systems perform different pre-processing tasks on their input data, requiring different amounts of time. For instance, a bi-gram system might only need to represent a document as a vector of two characters, while a concept-based system will need to go through a text-analysis procedure involving several steps before starting to perform the classification process itself.

Another factor to look at, especially in the case of machine learning systems, is the training time. Time to training a system can range from a few minutes to several days. Another factor to consider is the speed curve. If the curve is linear, this will be simple. But in most systems, elements such as the number of nodes in the taxonomy, the quantity of training data, or the length of documents will cause the speed curve to either plateau or increase exponentially at some threshold.

Finally, in terms of test environment, producing the same software/hardware environments, while essential, is not always feasible, especially when comparing several systems together.

2.3.3 Scalability

Because of the need to classify large amounts of documents, there is an increasing need for systems that can achieve higher classification accuracy while maintaining the ability to process large document collections (Meretakis et al. 2000). In many instances a tradeoff between accuracy and scalability has to be considered in industrial environments. Scalability is defined based on:

- the through put of the system: how many documents it can handle
- Number of nodes in the taxonomy it can handle
- Document sizes it can handle

In previous sections we saw that in industrial environments, scalability of systems is of extreme importance. A greatly accurate system that has a speed of one document per 50MS, but does not scale, will be of little use. In fact, most software companies (as well as their clients) prefer a slower and less accurate system to one that crashes or hangs.

2.3.4 Maintenance

Maintenance is mostly a software issue that does not need to be addressed here. However, maintaining a system also includes the ability to make it evolve. System evolution may happen at two levels: Model and data. A system that can evolve easily is one that will see constant improvements, getting better with time. Making the model of the system evolve requires a model that is not too complicated, and effects of modifications are easily measurable. Data evolution is an important issue, especially in production environments. If adding a single category to the system requires retraining, chances are there will be more resistance to that addition.

2.3.5 Ease of implementation and use

A last criterion for evaluation is how easy the system is to use and implement. Sometimes systems require very complex pre-processing that makes them almost impossible to implement. Other times, the use is so complicated that only the designer of the system can operate it.

In the next section I discuss the most common Machine Learning approaches used for ADC.

2.2 Machine Learning[6]

Machine Learning is a very broad field concerned with the development of methods and techniques allowing computers to learn specific tasks. Machine Learning applications are numerous ranging from natural language processing, to stock market analysis, passing by pattern recognition, medical diagnosis, fraud detection, DNA sequences classification, handwriting recognition, object recognition, playing games, and robot locomotion.

A great advantage of Machine Learning models is that the same model can be applied to a number of different tasks, with few modifications in the model itself.

In this paper, the focus is on a few Machine Learning models that have been used extensively for Text Classification. The chosen models are:

- Space Vector Model
- Naive Bayes
- kNN
- Latent semantic indexing
- support vector machines
- Neural Network

2.2.1 Space Vector Model

Space Vector Models were the first attempts at representing textual data in such a way that comparisons between documents could be made based on mathematical models. Although not usually recognized as a Machine Learning model in itself, it is the foundation of all the ML models seen in this paper.

[6] **Disclaimer**: I only have a working knowledge of different ML models, and a superficial knowledge of the internal workings of the models presented in this paper. The material presented here is the result of research done specifically for this paper, and some of the information may not be complete. As much as possible, I have cross checked the information for accuracy, but in some cases I have taken the claims of the authors at face value.

A space Vector Model is a vectorial representation of a textual document devised initially for information retrieval tasks (Salton, 1981) within SMART (Salton's Magic Automatic Retriever of Text) System. SMART used the concept of Space Vector Models to represent documents as vectors whose dimensions corresponded to the words of the document. It then represented a user's queries as the same kind of vector as the document vectors. Relevancy rankings of documents in a keyword search could then be calculated by comparing the deviation of angles between each document vector and the query vector. In a formula such as:

$$\mathbf{Cos\ \beta = (V1 \,.\, V2) / (|V1|\ |v2|)}$$

Where a cosine of 0 would mean that the query and document vector are orthogonal and therefore had no match, where as a cosine of 1 would mean that the document contained only the word that was also the user's query.

It is easy to see that this concept can be extended to machine learning by varying *user_query* to *document_to_class:* Instead of sending the system a vector containing a user query, we can just send it a vector containing the document to be classed. The documents that return on top (cosine near 1) are closest matches to the document that we want to class. If the base documents (training documents) contain category assignments, then we can assign those categories to the new document.

Given corpora of documents and a training set of examples of classified documents, the technique locates a minimal set of co-ordinate keywords to distinguish between classes of documents, reducing the dimensionality of the keyword vectors (Bao et al. 2000). However, some refinements are required, especially at the stage of category assignments, but we can see that the fundamental design of a Space Vector Model is radically different from the lexicon and rule-based models we saw earlier.

The limitations of this model for text classification include poor handling of long documents that match almost everything; too many false positives because of poor handling of word substrings;

loss of word order in documents resulting in less accuracy; and a problem with the fact that documents with similar subject matter but different vocabulary match poorly.
There are no formal evaluation benchmarks for this model in the context of text classification, and therefore it is not included in the final comparison table. However, given its fundamental role in mathematical models of NLP, it was highly relevant in this section.

2.2.2 Naive Bayes

Naive Bayes classifiers were among the first Machine Learning models developed for the task of text classification, and remain to this day one of the classifiers most used in both industry and academia due to their robustness, simplicity[7] and ability to produce relatively good results in many varying contexts (Lewis, 1998; McCallum and Nigam, 1998; McCallum and Nigam, 1998; Yang, 1999; Kim et al. 2000, Kibriya et al. 2004; Frasconi et al. 2001; Peng and Schuurmans, 2003; Yin and Power, 2006; He and Ding, 2007; Schneider, 2005).

This paper will provide a more comprehensive description of the inside workings of Naive Bayes classifiers to demonstrate how a simpler machine learning model is designed. Other systems will not be covered in as much detail.

Naive Bayes classifiers are based on the application of Bayes' probability theorem which relates the conditional and marginal probabilities of two random events to compute posterior probabilities given anterior observations:

$$\mathbf{P(C|X) = P(X|C)\,P(C) / P(X)}$$

Where :

P is the probability

X is the unknown

C is a possible outcome of X based on posterior observations

[7] It's been suggested that the "Naïve" part of Naïve Bayes is due to the inherent simplicity of this type of classifier. In fact, at least according to Yang and Lie (1999), "the naive part of NB methods is the assumption of word independence, i.e., the conditional probability of a word given a category is assumed to be independent from the conditional probabilities of other words given that category". But in fact it is this assumption that makes the computation of Naïve Bayes classifiers simpler than other approaches because it does not use word combinations as predictors.

P(C) is the prior probability of C

P(X) is the prior probability of X

P (C|X) is the posterior probability of C conditioned on X

P (X|C) is the posterior probability of X conditioned on C

Naive Bayes classifiers can handle an arbitrary number of independent variables whether continuous or categorical. Given a set of variables, X = {x1,x2,x...,xd}, we want to construct the posterior probability for the event Cj among a set of possible outcomes C = {c1,c2,c...,cd}. In a more familiar language, X is the predictor and C is the set of categorical levels present in the dependent variable. Using Bayes' rule:

$$P(C_j|X_1, X_2,....X_d) \propto p(X_1, X_2, ...X_d|C_j)p(C_j)$$

Basically, the theorem states that the probability of a future event is a function of both conditional and marginal probabilities. From this, the idea behind a Naive Bayes classifier model is to use the combined probabilities of words and categories (both represented as Space Vector Models) to estimate the probabilities of categories for a new document. The following description from Shen and Jiang (2003) summarises the construction of a classifier based on the Bayes Theorem:

Let Y = {0, 1, . . . , |Y|} be the set of possible labels for documents, and W= {w1 , w2 , . . . , w|W|} be a dictionary of words. A document of N words is then represented by the vector X = (X1 , X2 , . . . , XN) of length N . The Ith word in the document is Xi The multinomial Naive Bayes model assumes that the label Y is chosen from some prior distribution p(Y = ·), the length N is drawn from some distribution p(N = ·) independently of the label, and each word Xi is drawn independently from some distribution p(W = ·|Y) over the dictionary. Thus, we have:

$$p(X = x, Y = y) = p(Y = y)p(N = n)\prod_{i=1}^{n} p(W = x_i|Y = y)$$

Now let there be a training set $S = \{X^{(i)}, Y^{(i)}\}_{i=1}^{m}$ of m examples

The Naive Bayes classifier uses S to calculate estimates p(X|Y) and p(Y) of the probabilities p(X|Y) and p(Y). The estimated probabilities are typically calculated in a straightforward way from counts from training data. To classify a new document, it is straightforward to see using Bayes rule that the estimated class posterior probabilities are[8]:

$$\hat{p}(Y = y_0|X = x) = \frac{\hat{p}(X = x|Y = y_0)\hat{p}(Y = y_0)}{\sum_{y=1}^{|\mathcal{Y}|} \hat{p}(X = x|Y = y)p(Y = y)}$$

Where

$$\hat{p}(X = x|Y = y) = \prod_{i=1}^{n} \hat{p}(W = x_i|Y = y)$$

The predicted class for the document is then just: arg max y∈Y p(Y = y|X =x).

The classifier's model parameters are approximated with relative frequencies from the training set. There are numerous variations of the above model, some reporting important improvements in accuracy. However, none of the studies seem to be conclusive. "Smoothing" techniques are almost always present, and other techniques are used to overcome the model's pitfalls. For instance, if a given class and feature value never occur together in the training set then the frequency probability will be zero, which will make all the other probabilities zero as well when they are multiplied. To remedy this, a sample correction in all probability estimates is incorporated so that a probability is never set to be exactly zero.

Despite, or perhaps because of, their simplicity, Naive Bayes classifiers often perform extremely well in real world situations. Recently, careful analysis of these classifiers has shown that there are theoretical reasons for the apparent efficacy of Naive Bayes classifiers (Zhang, 2004). Several new Machine Learning approaches are based on the Naive Bayes model, and countless implementations of Naive Bayes using different feature sets, training parameters, and smoothing algorithms.

[8] Note that p(N=n) is dropped from the initial equation given length n of the document does not depend on the label.

Among the advantages of Naive Bayes classifiers we can mention the small amount of training data that it requires. For instance, for taxonomies in the range of a few dozen to a few hundred nodes, training data of around 10 documents per node are often used. In terms of scalability, Naive Bayes classifiers can handle several thousand categories with relative ease, although accuracy suffers as the number of nodes is increased. In terms of accuracy, several studies using the standard Reuters or similar corpora have reported F-measure rates in the low 80s, with precision being usually a few percentage points higher than recall. More information is provided in a later section.

2.2.3 Knn (K nearest Neighbor)

Knn was first introduced in 1951 in a paper called "Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties," by Fix and J. Hodges. Initially it was intended for the task of pattern recognition, but quickly it was adapted as a text classifier (Dasarathy, 1991). It has consistently been ranked as a top performer in standard evaluations, and is even simpler than the Naive Bayes model presented in the previous section. In fact, as will be seen, Knn closely follows the principals of the Space Vector Model described at the beginning of this section (Yavuz and Güvenir, 1998; Kwon and Lee, 2003; Han et al. 2001; Cardoso-Cachopo and Oliveira, 2003; Larkey, 1998; Liao and Vemuri, 2002; Zelikovitz and Hirsh, 2002).

Knn's basic algorithm is to find for a given test document the k nearest neighbours among the training documents, and use their categories to find the best categories for the test document. The "k nearest neighbours" are determined by a similar process as the Space Vector Model: each training document and test document are represented as a multidimensional feature space vector. The cosine value, or more commonly the Euclidean distance between the two vectors, is used to determine the degree of similarity. The classification is based on a "vote" of the neighbours, with the document being assigned to the classes most common amongst the nearest neighbours. The sharing of categories among K nearest neighbours results in the addition of the category weight, increasing the likelihood of that category to be assigned to the test document. Once all the category weights have been determined, a ranked list is obtained. A threshold is applied to

the scores, and only the strongest categories are conserved. The decision rule in kNN is written as (Yang, 1999):

$$y(\vec{x}, c_j) = \sum_{\vec{d_i} \in kNN} sim(\vec{x}, \vec{d_i}) y(\vec{d_i}, c_j) - b_j$$

Where:

$y(di, cj) \in \{o, 1\}$ is the classification for document di with respect to category cj

Sim (Xi, di) is the similarity between test document Xi and the training document di

Bj is the category specific threshold for the binary decision

The training phase of the model consists mainly of storing the feature vectors and class labels of the training samples, making the model quite simple in terms of both training and processing. Yet, as mentioned earlier, Knn has consistently proven to be a strong contender among text classifiers, scoring often close to or better than Naive Bayes. Because of simplicity, it is also a fast and robust classifier that is regularly implemented in industrial environments. A drawback to Knn is that the classes with the more frequent examples tend to dominate the prediction space, as they tend to come up more often in the K nearest neighbours. One way to overcome this problem within the model is to take into account the distance of each K nearest neighbours and predict the class of the new vector based on these distances. Another way is to analyse the training set in advance and reducing the number of documents containing the dominant classes. Another drawback of Knn is that it uses all document features in computing distances measure of similarity. It is believed that only a smaller number of the total vocabulary may be of interest for categorisation. One way to overcome this issue is creation of "weight-adjusted Knn" which assigns weights based on criteria such as ti-idf (Han et al, 2001). A last drawback of Knn is that relatively larger training sets are required for obtaining good results. Typically, a balanced corpus of around 25 to 50 documents per taxonomy node is used.

2.2.4 Latent Semantic Analysis

Latent Semantic Analysis (LSA), and Probabilistic Latent Semantic Analysis (PLSA) Machine Learning models were specifically designed for language processing tasks (Hull, 1994; Liddy et

al. 1994). The previous models that we looked at rely on vector representations based on explicit features from the text. An LSA also uses vectors to represent textual data, but LSA's vectors contain not only explicit features from documents, but also implicit features, called concepts, that do NOT come from the documents.

The basic idea behind LSA was developed following a series of experiments in vocabulary use by humans. It was observed that different people often use different words to convey the same ideas. Vocabulary variations between people seem so deeply rooted that even when people were directed to use certain words in specific context, they would mostly revert to their own way of saying things. These observations led to the hypothesis that textual documents produced by different people would also carry an individual's personal signature in terms of vocabulary use. It was therefore concluded that the very idea of "keyword searching" was destined to failure in many cases, because a user query would only match those documents whose authors use the same type of vocabulary as the user to convey the same ideas.

LSA was developed with the idea of modeling the data in such a way that in addition to the explicit features of the data set (usually words in the case of textual documents), the implicit concepts in the data set could also be exploited. In order to achieve this, the model uses a large training set. First, the number of words in the documents are severely reduced by removing all but "content" words. Then, documents and their words are represented in a matrix. Each row in the matrix is a vector corresponding to a term, and each column of the matrix is also a vector but corresponding to a document. The model then creates a "concept space" by comparing the words of each document against the rest of the collection to find which ones contain some of the same words. LSA considers documents with many words in common to be semantically related, while those with few words in common to be semantically unrelated (Esuli and Sebastiani, 2005; Ginter et al. 2005; Rosso et al. 2004; Blei et al. 2003; Cristianini et al. 2002; Ontrup and Ritter, 2001).

The concept space usually has thousands or even tens of thousands of dimensions. Each document in the collection is a vector with as many components as there are content words. LSA then reduces the gigantic multidimensional space into a much smaller vector by generalizing the

correlation between the terms over the documents by using Singular Value Decomposition (SVD) technique. SVD allows the representation of the words and documents in a reduced dimensional space, with similar words grouped together. Each group of similar words is assigned a concept tag which represents that group. The concepts are then represented as vectors as generalisations of the content words.

From this point on, the classification process is basically the same as in other models, except that a test document that goes through the system needs to be first translated into the concept space. The biggest advantage of this model is that by using the concept space, it can match documents that have few or even no words in common, but have the same *aboutness*.

This is visibly a very heavy duty system with important limitations and drawbacks, especially for the task of classification. For one thing, the foundation of the approach is based on generalising the data. This might be effective for information retrieval, but, in the case of classification, discriminating data is equally important, especially as the number of classes increases and there are more overlaps between the classes. The model also lacks in the areas of scalability and robustness. It has rarely been tested on taxonomies over a few hundred nodes, and already at this level it starts to show some signs of stress. The creation of that huge matrix in memory is also a source of important software issues. In terms of accuracy, the system can be extremely accurate with a few categories. For instance, in evaluations against the Reuters corpus, using only the top 10 nodes, F-measure ratings of 95% and more are not unusual. However, as the number of categories increase, the model's accuracy diminishes to levels below 70%.

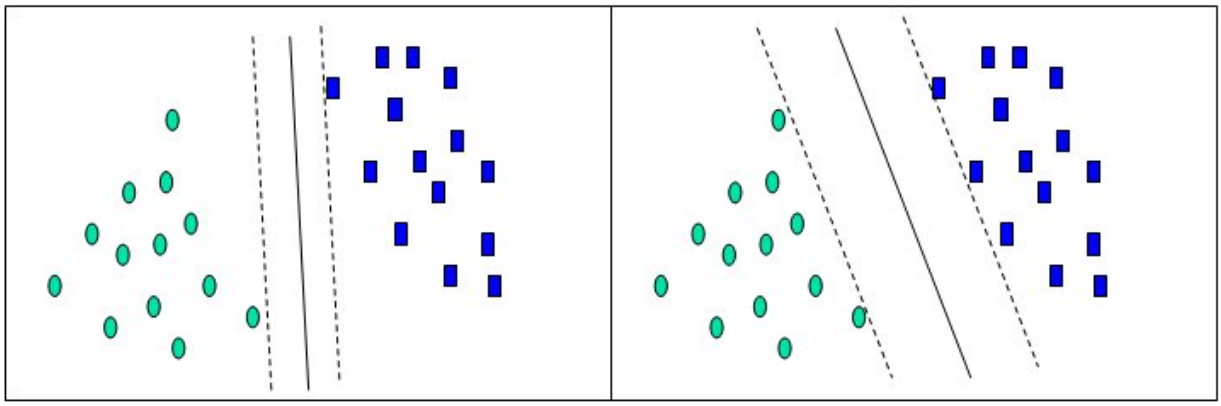
Overall, LSA is an interesting experiment in modeling linguistic data in novel ways. The model has had some great success stories, however, for the problem of text classification, more work needs to be done.

2.2.4 Support Vector Machine (SVM)

SVM, or support Vector Machine, was invented in 1979 by Vladimir Vapnik, however it become popular only in the mid 90's and has continued to become one of the major classification systems

both in industrial and academic settings (Kwok, 1998; Tong and Koller, 2000; Brank et al. 2002; Drucker et al. 1999; Klinkenberg Joachims, 2000; Zhang et al. 2003; Cai and Hofmann, 2004; Joachims, 2001; Joachims, 2002; Joachims, 1999; Kim et al. 2005; Shanahan and Roma, 2003; Wang and Son, 1999; Huffman and Damashek, 1994). SVM classifiers have the reputation of being fast, robust, and accurate.

SVM classifiers are also referred to as Maximum Margin classifiers because they simultaneously minimize the empirical classification error and maximize the geometric margin (Li and al, 2007)[9]. SVM performs classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories. The goal of SVM modeling is to find the optimal hyperplane that separates clusters of vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other size of the plane. The vectors near the hyperplane are the support vectors, thus the name Support Vector Machine. To illustrate, consider the following example:

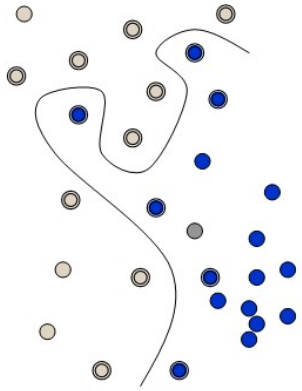


Data points from two categories (blue rectangles and green ovals) are plotted along the X and Y axis. The line on both figures has separated the two categories, however the separation on the right side is optimal given the larger margin between the solid and dotted lines. We notice that the dotted lines on the right figure are using the data points as supports to maximize the geometric margin.

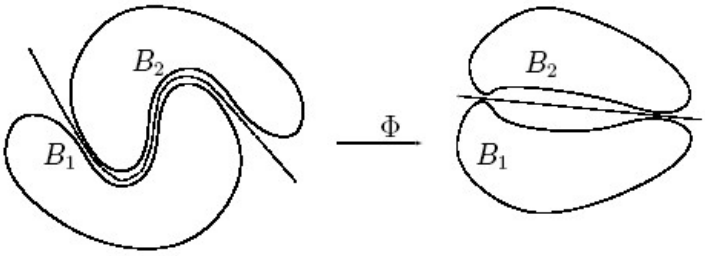
SVM is of course capable of handling more than two categories, as well as separating data by non-linear curves, and even handling of cases where data clusters cannot be separated.

[9] The following section, especially the diagrams, is largely from : http://www.dtreg.com/svm.htm

One of the “beauties” of SVM is how it handles non-linear curves. Consider the data set below:

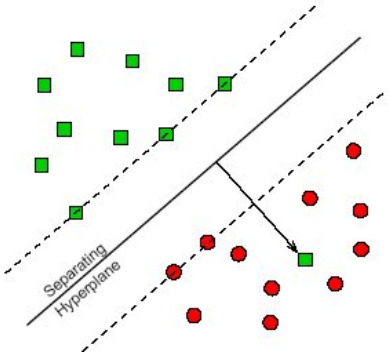


Rather than fitting a nonlinear curve to the data, SVM handles this case by using a kernel function to map the data into a different space where a hyperplane can be used to do the separation:



The concept of a kernel mapping allows SVM models to perform separations even with very complex boundaries. Different kernel functions are possible : linear, polynomial, radial basis, and sigmoid.

Another characteristic of SVM is how it deals with non-separable trainings data as in the case below:



In this case, SVM, will calculate the trade off between accepting the error or forcing more rigid margins. It creates a soft margin that permits some misclassifications in exchange of flexible margins that generalize much more easily.

Training an SVM requires the solution of a large quadratic programming (QP) optimization problem. There are several ways to train a SVM classifier. A fast method is called Sequential Minimal Optimization (SMO). SMO breaks this large QP problem into a series of smallest possible QP problems. These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop. The amount of memory required for SMO is linear in the training set size, which allows SMO to handle very large training sets.

Overall, SVM has become the classifier of choice for many. In terms of accuracy, it is often seen at the top of the chart. As we have seen, it is one of the more elegant models. In terms of robustness and speed, it is also one of the better classifiers. However, because of the model's design, it does not scale very well. Training time increases very fast with the increase of the number of categories. A few years ago, some Microsoft researchers found methods to partially overcome this drawback, enabling SVM to handle up to a few thousand categories. The last drawback of SVM is that adding a new category into the vector space requires retraining the whole system.

2.2.5 Neural Networks (NN)

SVM models are closely related to Neural Networks. In fact, a SVM model using a sigmoid kernel function is equivalent to a two-layer perceptron neural network. Many "flavours" of NN

have been developed and tested. A few examples include: "Adaptive Incremental Learning" (Geutner et al. 1993) which learns interactively to classify documents, eliminating batch training runs ; "Hierarchical Mixture of Experts Model (Ruiz and Srinivasan, 2002) which uses a divide and conquer principle to define smaller categorization problems based on a predefined hierarchical structure; Principal Component Analysis (PCA) to select the most relevant features for the classification (Selamat and Omatu, 2004) and then combining it with the feature vectors from the class-profile; and many others (Wermter et al. 1999; Geutner et al. 1993; Li et al. 1991; Wiener et al. 1995; Koehn, 2002; Lee et al. 2003; Liu and Zhang, 2001; Tan, 2001; Ginter et al. 2005).

All the ML models that were covered in the previous sections are strictly supervised learning system. Neural Networks, according to the literature, can be implemented as both supervised or unsupervised. As a supervised learning system, Neural Networks have been the object of numerous evaluations. Wiener et al (1995) evaluated NN against the standard Reuters corpus with both the perceptron approach (without a hidden layer) and three-layered neural networks (with a hidden layer). Both systems use a separate neural network per category, learning a non-linear mapping from input words (or more complex features such as singular vectors of a document space) to a category. Wiener's experiments suggested some advantage for combining a multiple-class NN (for higher-level categories) and many two-class networks (for lowest-level categories).

Yang et al (1999) conducted another series of experiments with NN using the same corpus. They modified the training parameter and trained one NN on all 90 categories instead of training the categories one by one. They used a hidden layer with k nodes, where k was empirically chosen. They also implemented their own NN system for efficient handling of sparse document vectors. Their objective was to compare 5 classifiers on the same data and conditions. They ran one set of tests with a reduced training corpus, and a second one with the full corpus. The results obtained are summarized in the following table and graphs:

Global Scores

method	miR	miP	miF1	maF1	error
SVM	.8120	.9137	.8599	.5251	.00365
KNN	.8339	.8807	.8567	.5242	.00385
LSF	.8507	.8489	.8498	.5008	.00414
NN	.7842	.8785	.8287	.3765	.00447
NB	.7688	.8245	.7956	.3886	.00544

miR = micro-avg recall; miP = micro-avg prec.; miF1 = micro-avg F1; maF1 = macro-avg F1.

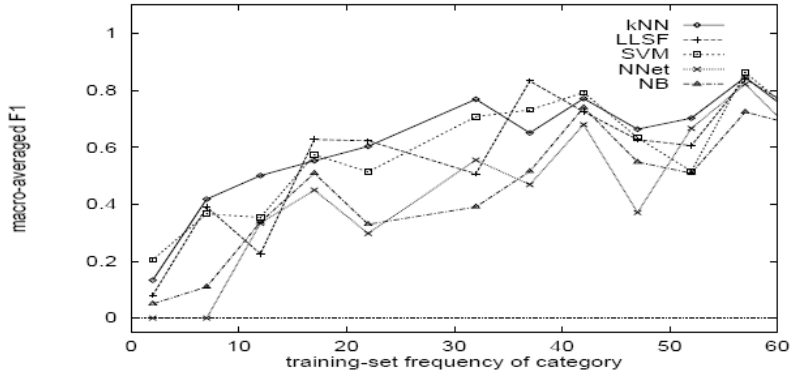


Figure 4: Performance curves on rare categories.

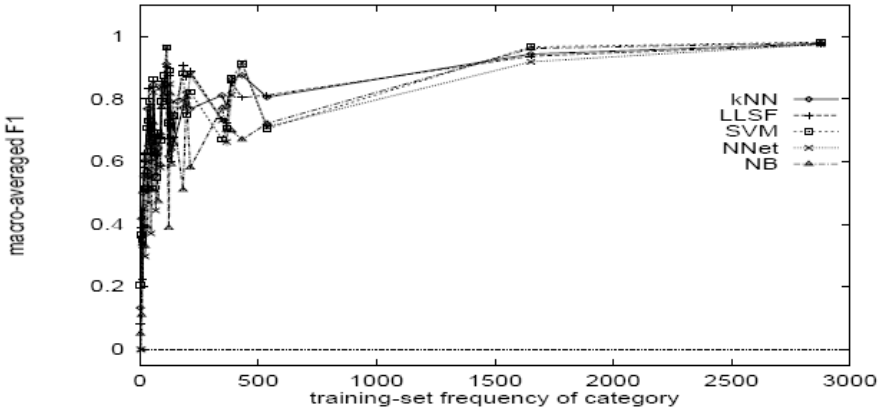


Figure 5: Performance curves on all the categories.

Figures 4 and 5 compare the performance curves of the five classifiers with respect to the training-set frequency of categories. Figure 4 focuses on the training set frequency in the range from 1 to 60 (covering 67% of the total unique categories), where NN and NB are clearly worse than the other three, but these three are less easy to rank. Figure 5 shows the performance curves on the full range of training-set frequencies of categories, where the effectiveness of all the classifiers are more similar to each other on common categories (with a training-set frequency of 300 or higher), compared to their relative performance on rare categories.

These results indicate that there are no perfect classifiers out there. Accuracy rates in the range of 85%, while respectable, do not fully satisfy the practical needs of the industry. It should also be noted that even the very rudimentary lexicon-based systems that were discussed in this paper, normally have accuracy rates in the neighborhood of 70-75%, using automatically generated dictionaries. With hand crafted dictionaries for a corpus such as Reuters classed in less than 100 nodes, obtaining 85% is not an impossible task. Granted, a lexicon-based system requires much more effort to implement, maintain, and has other shortcomings that were already discussed. However, this tells us that despite the relative maturity of machine learning systems, they are still performing at a level more or less comparable to the more rudimentary lexicon and rule-based systems. So the question to ponder is what is missing in the equation and what should be done to make significant gains in this domain?

The rest of this report does not aim to explore the above question but rather will focus on alternative means of using the existing classification systems to enhance content management within organizations.

3. Facetted Classification

Facetted classifications offer multiple views on the same object thus enabling users to retrieve the information in different ways. They are particularly useful in work environments where the same information is likely to be of interest by multiple stakeholders working in different sectors with varying interests. An example of such an environment are government organizations. Facetted classifications, if designed to be intuitive and comprehensive, provide a powerful means for organizing information.

Implementation and use of facetted classification systems can be challenging. At a very high level, facets need to handle three or more dimensions of classification. Facets must be mutually exclusive and jointly exhaustive. They should be flexible, expressive, expandable, allow different perspectives and have the ability to accommodate new entities rather easily. Choosing the right facets, establishing comprehensive relationships between them and the multidimensional visualization of facets are among common problems one can encounter when creating facetted classification systems.

Compound terms in languages such as English and French have an exponentially higher expressive power than single words. They often have a single referent and thus circumvent the problems related to polysemy. In addition, through their decomposition, certain semantic relationships can be established without the use of additional databases. While the use of compound terms in facetted classification systems can seem advantageous, an efficient method of implementation needs to be devised at the design stage. The method must particularly take into account the multidimensionality of facetted classification, the relationships between facets and means of visualization.

The rest of this report will outline methodological criteria for putting in place a facetted classification model that uses compound terms to dynamically represent and retrieve specific informational content from unstructured textual data. A robust methodology requires the identification and the use of the right tools and resources, formalization of a process that can be automated to a large degree and conducting a proof of concept that demonstrates the feasibility

of the method. While not all these aspects could be fully covered in this project, a conceptual model is presented which can form the bases of a more comprehensive project.

4. Compound Terms

Syntax-rich languages such as English and French use word combinations extensively to create compound terms. A compound term (e.g. Public Service; Lieutenant Colonel; Government of Canada; Mayor-Elect, Blackboard, etc.) refers to a unique concept. The grammatical structure and semantic structure of compound terms are quite varied. As a general rule, compound terms are formed to narrow down the semantic range of a more general concept. Some examples include: civil servant (type of job function), recycling bin (purpose of an object), immersion heater (mode of a function), night frost (when something happens), doorbell (where something is), wood cabinet (what something is made of), etc. One of the aims of this report will be to show how the syntactic and semantic structures of compound terms can be manipulated to pinpoint contextual meanings that they convey not only in a single document, but more importantly across the set of documents in an organization.

Compound terms have been the subject of numerous studies in the past and different methods have been developed to identify and extract them. The two most common methods are statistical and linguistic based. For the purposes of this report, we will assume that the extraction of compounds from a corpus is a given. Throughout this project, the term extraction system created by TerminoWeb (http://terminoweb.iit.nrc.ca) has been used to conduct the experiments.

The project's focus has been on developing an approach that relies on the contextual use of compound terms in an organization's documents. It is assumed that a large set of documents (corpus) is available for processing, and that the corpus is a good representation of the main knowledge areas of the organization.

Various methods of manipulating compound terms were explored to identify and establish syntactic and semantic links between compounds. The following three methods have proved to be the most productive and are also used in the TerminoWeb:

1) Conceptual relationships – This is a straightforward but highly productive method. It consists of using the common elements in compounds to group them together. For example, *video card* and *sound card* share the element *card* and can be linked together by this simple method. However, many false positives can also be expected (e.g. *library card*).
2) Contextual relationships – This is also a straightforward method but requires large corpora to function well as it relies on probabilities of co-occurrence between compounds. The method consists of parsing a corpus for the "neighbours" of each compound. The more often two compounds occur near each other the more likely they have a semantic relationship. For example, *video card* and *sound card* have a much higher chance of being used in the same contexts than with *library card*. Combined with the first method it can be used to purge occurrences such as library card in addition to yielding new connections with compounds such as computer hardware. It should be noted that the method can be used iteratively by calculating the "neighbours of the neighbours". However the iterations can become computationally intensive very quickly.
3) Syntagmatic relationships: Much less productive but more precise, this method consists of identifying a number of "indicators" or patterns that occur regularly in language. "is a" is a typical indicator indicating membership as in "a video card is a computer component". TerminoWeb project has created a few dozen different classes of syntagmatic relationships and has an open architecture to add new classes, and new indicators to each class.

The combination of these methods and their application to large corpora is likely to yield interesting results for the linking of compound terms, and therefore the concepts used in an organization's documents, specifically for the dynamic creation of facetted taxonomies. Once the links have been established, compound terms can be clustered into groups that represent areas of knowledge in an organization.

Given constraints of time and resources, a large scale experiment was not possible in this project and therefore the focus was placed on creating a basic model that may serve as a blueprint for a large scale project. The ultimate goal of this report is to outline this model and to show how such

a model may contribute to the successful auto-classification of documents against a multi-faceted classification scheme based on the content of textual documents in organizations.

5. Information Representation and Retrieval

The model presented in this report has two main aims which are not separable: Information representation and information retrieval. It is not meaningful to separate these two process since each serves and depends on the other. While the end goal is retrieving information from content, this goal cannot be achieved without an adequate information representation. Similarly, information representation without having information retrieval as an end goal is meaningless and does not any purpose. Information representation and information retrieval are therefore the two sides of the same coin and need to be considered jointly. However, in designing models and systems, they need to be considered as two separate, but related, process.

A relatively simple, yet powerful, information retrieval system is the search engine. Search engines represent the textual data in documents in the form of a search index. The search index represents in a highly compact format the contents of a corpus which can contain millions, even billions, of documents. A search query sent by the user triggers the search engine to search through the search index and return the list of pertinent documents. This process is enhanced by sorting algorithms which place documents that are more pertinent to search query on top of the list.

While fast, inexpensive and powerful, search engines suffer from several draw backs, notably poor precision: List of retrieved documents contains a high percentage of irrelevant documents and the researched item may be buried way down the list making it difficult to retrieve. Poor recall is another problem since sometimes the document that is searched for may not contain the words of the query and therefore will not be retrieved at all. In addition to these two basic problems, search engines cannot discriminate between content types. For example, if an information retrieval process only has interest in scientific articles or court judgments, a search engine will not be able to return only these kinds of documents, unless this information has also been indexed, which is not the case with general search engines today. Another example of

content type indiscrimination has to do with the meaning of words in different contexts. An often used example is the word *apple* which can refer to the company Apple or the fruit (among several other things). Countless number of other examples can be cited and are experienced on a daily bases by anyone who uses search engines.

It is the assumption in this report that addressing this third problem will substantially alleviate the first two problems (precision and recall) as well, making information retrieval highly more efficient and powerful.

To address the problem of content type, some search engines add metadata to documents and index the information along with its contents. For example, Google identifies document formats such as PDF, Word, Powerpoint, Excel, etc. and allows users to limit their search to only these kinds of documents. Similarly, Google News tags news articles by subject matters such as Health, Business, Sports, etc. and allows its users to search only those kinds of documents.

One of the underlying ideas presented in this report builds on this idea of adding meta-data to content of documents, but at a more granular level. In addition, the meta-tagging is envisioned to be a dynamic process which can be changed based on the usage context. The model allows:

- Contextualizing of search
- Navigation of results by different angles
- Segmentation of search results by topic or subject
- Dynamic clustering of search results based on needs
- Interactive searching using different metadata types such as:
    - **People**
    - First Name
    - Last Name
    - Address
    - Zip / postal code
    - Phone number
    - Email address
    - Job title
    - Place of work (ON)
    - Social security / insurance number
    - Nationalities

- **Geographical Locations**
- Country
- City
- State / province
- Streets
- Others (lakes, rivers, mountains, etc.)
- **Organization names**
- Company names
- Ticker symbols
- Location
- URL
- Financial Information (revenue, profits, etc.)
- Industry codes
- ISO Standards
- **Measures**
- Currency
- Date
- Day / Month / Year
- Percent
- Time
- Time period
- **Science, Technical, Medical (STM)**
- Chemical names
- Organisms
- Drugs
- Diseases
- Anatomy
- Genes
-
- **Event names**
- Holidays
- Sports
- Political (Elections, Referendum, conferences)
- Cultural (Festivals, Expositions, etc.)
- Scientific (summits, Congress, etc.)
- Historic (Wars, Treaties, etc.)
- **Products**
- Vehicles (model/color/year)
- Aviation
- Electronics
    - Computer Hardware
    - Computer Software
    - Home Entertainment (TVs, Audio, etc)
    - Small Electronics (players, cameras, gadgets)

    - Telecommunications
- Sports (Equipment)
- Home Appliances
- Weapons
- Toys
- **Entertainment**
- Musicians (Bands and Artists)
- Song and album titles
- Books (Titles, Authors Publishing house)
- Movies titles
- Actors, Actresses, Comedians
- T.V. programs
- T.V. and radio personalities
- Athletes
- **Classified Ads**
- Homes (size, price, location)
- Cars (Mileage, no. doors & cylinders, transmission type, speed, make, year, price)

The above scheme, while partial, represents some common metadata types that can be used to pinpoint precise information types in various contexts. It should be noted that this scheme is itself a simple taxonomy which can be combined with other taxonomies to further augment precision.

While we have discussed the above in the context of search engines, another way of looking at the use of taxonomies for information retrieval is the direct use of taxonomies by a process known as information browsing.

Browsing is the process of drilling down information through hierarchical structures. The browsing metaphor is used extensively for structured data. For example, computer systems have a hierarchy of folders which contain files (as leaves). Typically, the root of the hierarchy is a hard drive which contains a number of folders, and possibly some documents. Each folder contains other folders, and possibly some documents, and so on. Sometimes the depth of the hierarchy can be a dozen or more levels. Regardless, the logical structure can be navigated easily if the user knows the logic behind the structure.

When dealing with textual content from large corpora, this strategy is much harder to implement and use. The main reasons have to do with the sheer vastness of possible structures and the varying logic behind the structures. The number of first level folders of a typical hard drive is usually a few dozen or less. If this number increased into hundreds or thousands, then the first level list would be too over whelming to browse through. Similarly, if the depth of nodes increased beyond a few dozen, the process of getting to the leaves of the nodes would require too much effort making the process unusable.

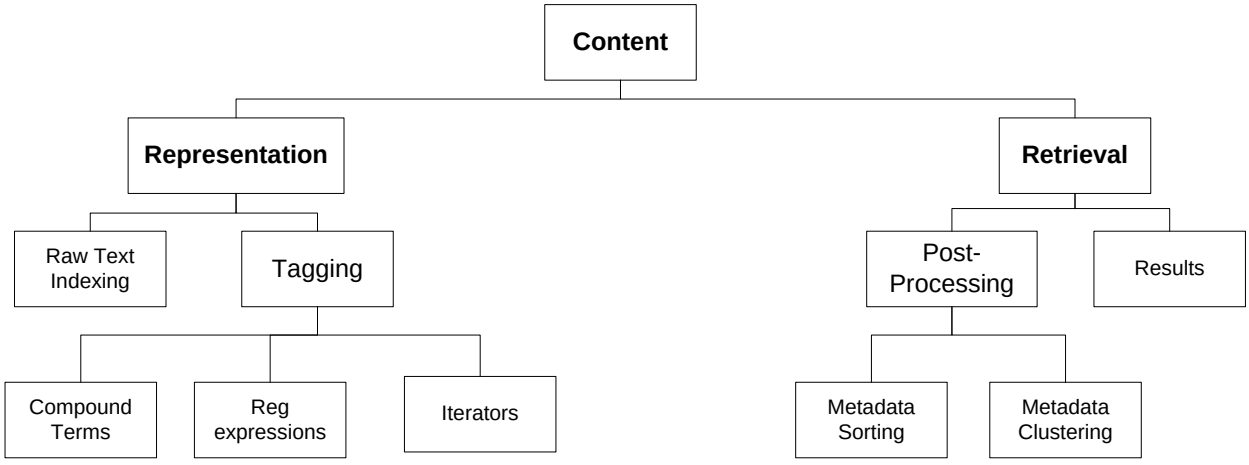
Having said this, the basic metaphor of browsing remains a useful and intuitive process and should not be completely discarded. In the model presented in this report, this problem of vastness of potential nodes is addressed through the dynamic construction of the taxonomy which will limit its breadth and depth considerably.

The second problem, that of the logic behind the structure, is one that librarians and information managers have tried to address for decades. While some standards exist in certain environments such as a library, in most cases the “logic” behind any content structure is for the most part a function of how the information manager envisions the information structure of his or her organization. While good reasoning and logic can be behind this vision, it is often the case that the vision is not shared by others who may see the structure as arbitrary for the most part. Documentation and training have had some beneficial effects but have proved extremely expensive and time consuming. Furthermore, as new content is added to or removed from a repository, the logic may shift. Finally, it has been shown that even those who conceive of information structures can get lost in their own schemes after a time has lapsed: what seemed perfectly logical in a given context and time, may seem arbitrary in another context and time. For these reasons, this model proposes the ability to create logical structures on the fly, based on the vision of the end user within the specific context of use.

This discussion highlights both the strength and weaknesses of search and information browsing for the purpose of information retrieval. In the next section, the model proposed in this report will show how combining the strengths of both these processes can address a number of the weaknesses inherent in each.

6. The Model

At a high level, the model can be schematically represented as follows:



At the top, Content, refers to the set of corpora within an organization, usually unstructured data, but it is easy to envision combining both structured and unstructured data for a more comprehensive model.

On the left had side, Representation refers to the set of processes required to represent the content in an adequate way. There are two main components here: Indexing and tagging. The indexing process is similar to what was discussed in the previous section. The aim of this component in the model is to provide a data representation method that is fast, efficient, and powerful, allowing rapid (albeit more brute) access to content. The tagging component is what differentiates this model from typical search engines. It is here that various taxonomic components are integrated and enrich the representation process many fold. More discussion on this will follow.

The right hand side of the schema, Retrieval, outlines different retrieval components. Here as well, we have two main components: Results and Post Processing. The Results component is similar to typical search engine in that it contains a list of documents sorted by ranking algorithm. The algorithm is of less importance and can follow the simple tf-idf model presented in the Litterature review section. The reason is that the Tagging process has potentially improved precision many folds already, and the Post-Processing component can be used to further refine the results. The Post-processing component in this model is a set of "tools" used by the user to refine results of a search query. It is an essential component in this model which differentiates it from a typical search engine, and will be discussed in more detail below.

The basic idea behind this model is to combine search and browsing in an interactive and dynamic manner. Enriching information representation through tagging is essentially performed through the extraction and tagging of compound terms. Compound terms in documents are extracted and tagged as such. Additional information is added to compound terms by the use of two of the three processes described earlier: Contextual Relationships and Syntagmatic Relationships. The third process, Conceptual Relationships, is used in the Retrieval process in this model.

In addition, the tagging of compound terms can be enhanced through the use of and "entity identifier" which recognizes the nature of a compound terms in its context. For example, the compound term Government of Canada can be recognized as the name of an organization. Various academic and industrial tools are available for such tasks.

The tagging of compound terms using the processes mentioned above at this stage aims to establish semantic relationships between compound terms. All such processes are envisioned to be handled through rule-based or lexicon based engines, with iterations between processes to refine and reduce errors, by statistical methods. For example, the compound term Government of Canada maybe associated with a number of neighboring terms after processing a single document. These associations can be further refined when the same term is extracted from other documents and the same processes are applied. Once a full corpus of adequate volume is processed, statistically significant patters can emerge allowing the attribution of weights (or

other means) to each association. Similarly, if the “entity identifier” incorrectly tags Government of Canada as a Geographical Location in one document, but correctly in many others, through the comparison of all instances a stronger Organization Name tag can be associated to this term. It should be noted that eliminating any instances automatically is not recommended here as the system will allow in the Retrieval process to highlight these cases and possibly correct them.

What’s more, when new documents are added to the repository, they can be tagged iteratively using the “knowledge” stored through the tagging of the previous documents.

It should be noted that the tagging module does not need, and should not, be limited to compound terms, although this is the focus of this project. Tagging can be extended to many categories of concepts within the document (for example extraction and identification of dates, measures, document components such as title, keywords, etc.) and the document itself if a topic-based tagging is available. The conception of the model in the retrieval phase is that all such information can be used during retrieval as will be discussed next.

The Retrieval component of the model is where dynamic result handling takes place. We have discussed in some detail facetted taxonomies in previous sections. Each of the tagging processes discussed above can be considered a classification schema, the in core of which are compound terms. Given the enormous number of tagging possibilities, and the open architecture nature of this model, the taxonomies themselves should not be exposed to users except on demand. Once a user obtains the results of a search query, the Post-Processing component can expose relative concepts to the result set allowing the user to perform various tasks, the most important of which are browsing through results using compound terms.

To illustrate this idea, consider the example of a user who starts by a simple search query such as “Canada” then the Post Processing module will immediately use the Conceptual Relationships process to generate a list of compound terms that contain Canada, such as Government of Canada. In addition, other tags can be exposed to the user to allow the user to either refine the search or browse through other documents using related compound terms. In essence, the Post-Processing component generates several taxonomies based on the result set each time a search is

performed. This allows the user to view through a hierarchical schema the contents of the result set. It should be noted that no taxonomies were necessary to define, and all the nodes of the taxonomies were generated using compound terms extracted from the result set.

The Post-processing component has two sub-components for sorting and clustering of metadata. Metadata in this context is essentially composed of compound terms, but can also be added metadata such as the ones discussed earlier in relation to “entity identifiers”. In addition to browsing through system generated hierarchies which are based on the Syntagmatic and Contextual Relationships processes, metadata sorting and clustering would allow to regenerate the hierarchies based on other criteria, namely clustering and sorting. The clustering involves the division of compound terms in logical clusters which are envisioned on the fly by the user. For example, if related terms to Canada is highly extensive, the user can use another facet of the tagging scheme to group together all related terms that are also Organization Names. Conceptually, the new facet intersects with the available results in order to significantly narrow down the list of compound terms. Sorting allows another process which is similar but may be used for other kinds of information and intelligence retrieval needs. By intersecting different facets of the tagging scheme the user can quickly see how strongly, or weakly, different tags are related to the set of compound terms from a search query. For example, using the above search scenario, the user can sort all the tags to see that the search query Canada is more strongly related to a topic tag such as Tourism than to Organization Names.

A final note of great importance in this model is that once content is represented through granular tagging as presented above, search queries can become a lot stronger, resulting in more refined searching to begin with. For example, if the user knows in advance that his results are within documents that talk about Canada within the context of organizations, then the query can be formulated as [Canada: ORG] (assuming an ad-hoc query syntax here) to only retrieve and/or sort documents based on how the document tags refer to Organizations. This idea can be extended to include compound terms, even those not included in a document. If the Government of Canada is associated to the compound term Tourism, a query to retrieve documents that contain Government of Canada within the tourism context can return documents that contain neither term, but are still about these concepts.

7. Conclusion

Metadata and taxonomies can be used to facilitate the storage, organization, access to and retrieval of information. Metadata is data about data, information that accompanies a data element to provide information about its author, creation date, data type, subject matter or any other characteristic of interest or significance. A taxonomy is a classification, often hierarchical, of information components and their interrelationships that supports the discovery of and access to information. Metadata and taxonomies can work together to identify information and its features, then organize it for access, navigation and retrieval.

The model presented in this project sees metadata and taxonomies as complementary constructs that can be dynamically generated based on usage needs. That is, metadata may be used to create the taxonomy or its structure, or to label nodes and their content. Conversely, in a general sense, a taxonomy is seen as metadata because it specifies how an entire body of content is organized and how its components are related.

In this model, taxonomy nodes are created using compound terms. Compound terms are the lowest level of metadata which when associated with other compound terms can create powerful representation and retrieval paradigms. A document or other content may be attributed with several compound terms which together, once associated, can become taxonomy nodes.

The model is essentially conceived as an information retrieval mechanism, but may be used for other related purposes such as linking content and showing relationships between pieces of content in a topic map or ontology.

Using compound term generated taxonomy nodes as metadata elements to improve searches and navigational browsing can be envisioned for medium and large organizations which have critical needs in information management and see a return on invest in putting together the required infrastructure.

References

Avancini, H., A. Lavelli, et al. (2003). Expanding Domain-Specific Lexicons by Term Categorization. Proceedings of SAC-03, 18th ACM Symposium on Applied Computing, ACM Press, New York, US.

Bao, Y., S. Aoyama, et al. (2001). A Rough Set-Based Hybrid Method to Text Categorization. Proceedings of WISE-01, 2nd International Conference on Web Information Systems Engineering, IEEE Computer Society Press, Los Alamitos, US.

Brank, J., M. Grobelnik, et al. (2002). Feature selection using support vector machines. Proceedings of the 3rd International Conference on Data Mining Methods and Databases for Engineering, Finance, and Other Fields.

Cai, L. and T. Hofmann (2004). Hierarchical Document Categorization with Support Vector Machines. Proceedings of CIKM-04, 13th ACM International Conference on Information and Knowledge Management, ACM Press, New York, US.

Cardoso-Cachopo, A. and A. L. Oliveira (2003). An Empirical Comparison of Text Categorization Methods. Proceedings of SPIRE-03, 10th International Symposium on String Processing and Information Retrieval, Springer Verlag, Heidelberg, DE.

Cristianini, N., J. Shawe-Taylor, et al. (2002). "Latent Semantic Kernels." Journal of Intelligent Information Systems **18**(2/3).

Drucker, H., V. Vapnik, et al. (1999). "Support vector machines for spam categorization." IEEE Transactions on Neural Networks **10**(5).

Esuli, A. and F. Sebastiani (2005). Determining the semantic orientation of terms through gloss classification. Proceedings of the 14th {ACM} international Conference on Information and Knowledge Management, ACM Press.

Frasconi, P., G. Soda, et al. (2001). Text Categorization for Multi-page Documents: A Hybrid Naive Bayes HMM Approach. Proceedings of JCDL, 1st ACM-IEEE Joint Conference on Digital Libraries, IEEE Computer Society Press, Los Alamitos, US.

Geutner, P., U. Bodenhausen, et al. (1993). Flexibility Through Incremental Learning: Neural Networks for Text Categorization. Proceedings of WCNN-93, World Congress on Neural Networks.

Ginter, F., S. Pyysalo, et al. (2005). Document Classification Using Semantic Networks with An Adaptive Similarity Measure. Recent Advances in Natural Language Processing.

Ginter, F., S. Pyysalo, et al. (2005). Document Classification Using Semantic Networks with An Adaptive Similarity Measure. Recent Advances in Natural Language Processing.

Han, E.-H., G. Karypis, et al. (2001). Text Categorization Using Weight-Adjusted k-Nearest Neighbor Classification. Proceedings of PAKDD-01, 5th Pacific-Asia Conferenece on Knowledge Discovery and Data Mining, Springer Verlag, Heidelberg, DE.

He, F. and X. Ding (2007). Improving Naive Bayes Text Classifier Using Smoothing Methods. Proceedings of the 29th European Conference on Information Retrieval.

Huffman, S. and M. Damashek (1994). Acquaintance: A Novel Vector-Space N-Gram Technique for Document Categorization. Proceedings of TREC-3, 3rd Text Retrieval Conference, National Institute of Standards and Technology, Gaithersburg, US.

Hull, D. A. (1994). Improving text retrieval for the routing problem using latent semantic indexing. Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval, Springer Verlag, Heidelberg, Germany.

Joachims, T. (1999). Transductive Inference for Text Classification using Support Vector Machines. Proceedings of ICML-99, 16th International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, US.

Joachims, T. (2001). A Statistical Learning Model of Text Classification with Support Vector Machines. Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval, ACM Press, New York, US.

Joachims., T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. European Conference on Machine Learning (ECML).

Kibriya, A. M., E. Frank, et al. (2004). Multinomial Naive Bayes for Text Categorization Revisited. Proceedings of AI-04, 17th Australian Joint Conference on Artificial Intelligence, Springer-Verlag.

Kim, Y.-H., S.-Y. Hahn, et al. (2000). Text filtering by boosting naive Bayes classifiers. Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval, ACM Press, New York, US.

Klinkenberg, R. and T. Joachims (2000). Detecting concept drift with support vector machines. Proceedings of ICML-00, 17th International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, US.

Koehn, P. (2002). Combining Multiclass Maximum Entropy Text Classifiers with Neural Network Voting. Proceedings of PorTAL-02, 3rd International Conference on Advances in Natural Language Processing.

Kwok, J. T. (1998). Automated text categorization using support vector machine. Proceedings of ICONIP'98, 5th International Conference on Neural Information Processing.

Kwon, O.-W. and J.-H. Lee (2003). "Text categorization based on k-nearest neighbor approach for Web site classification." Information Processing and Management **39**(1).

Larkey, L. S. (1998). Automatic essay grading using text categorization techniques. Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval, ACM Press, New York, US.

Lee, H.-M., C.-M. Chen, et al. (2003). A neural network document classifier with linguistic feature selection. Proceedings of IEA/AIE-00, 13th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. Proceedings of ECML-98, 10th European Conference on Machine Learning, Springer Verlag, Heidelberg, DE.

Li, W., B. Lee, et al. (1991). Text classification by a neural network. Proceedings of the 23rd Annual Summer Computer Simulation Conference.

Liao, Y. and V. R. Vemuri (2002). Using Text Categorization Techniques for Intrusion Detection. Proceedings of the 11th USENIX Security Symposium.

Liddy, E. D., W. Paik, et al. (1994). "Text categorization for multiple users based on semantic features from a machine-readable dictionary." ACM Transactions on Information Systems **12**(3).

Liu, Z.-Q. and Y.-J. Zhang (2001). "A competitive neural network approach to web-page categorization." International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **9**(6).

McCallum, A. and K. Nigam (1998). A comparison of event models for Naive Bayes text classification. Proceedings of AAAI-98, Workshop on Learning for Text Categorization.

Mladeni\'c, D. (1998). Turning \sc Yahoo!\ into an automatic Web page classifier. Proceedings of ECAI-98, 13th European Conference on Artificial Intelligence, John Wiley and Sons, Chichester, UK.

Mladeni\'c, D., J. Brank, et al. (2004). Feature selection using linear classifier weights: interaction with classification models. Proceedings of SIGIR-04, 27th ACM International Conference on Research and Development in Information Retrieval, ACM Press, New York, US.

Ontrup, J. o. and H. Ritter (2001). Text Categorization and Semantic Browsing with Self-Organizing Maps on Non-Euclidean Spaces. Proceedings of PKDD-01, 5th European Conference on Principles and Practice of Knowledge Discovery in Databases, Springer Verlag, Heidelberg, DE.

Rijsbergen, C. J. (1979). *Information Retireval*. London, Butterworths.

Rosso, P., A. Molina, et al. (2004). Information Retrieval and Text Categorization with Semantic Indexing. Proceedings of CICLING-04, 5th International Conference on Computational Linguistics and Intelligent Text Processing, Springer Verlag, Heidelberg, DE.

Salton, G., McGill, M. J. (1983). Introduction to modern information retrieval, McGraw-Hill.

Salton, G. a. M., M. J. (1991). "Developments in Automatic Text Retrieval." Science **253**(5023).

Schneider, K.-M. (2005). Techniques for Improving the Performance of Naive Bayes for Text Classification. Computational Linguistics and Intelligent Text Processing.

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

Shanahan, J. G. and N. Roma (2003). Boosting support vector machines for text classification through parameter-free threshold relaxation. Proceedings of CIKM-03, 12th ACM International Conference on Information and Knowledge Management, ACM Press, New York, US.

Tong, S. and D. Koller (2000). Support Vector Machine Active Learning with Applications to Text Classification. Proceedings of ICML-00, 17th International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, US.

Wang, K., S. Zhou, et al. (1999). Building hierarchical classifiers using class proximity. Proceedings of VLDB-99, 25th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers, San Francisco, US.

Wermter, S. (2000). "Neural Network Agents for Learning Semantic Text Classification." Information Retrieval **3**(2).

Wermter, S., G. Arevian, et al. (1999). Recurrent Neural Network Learning for Text Routing. Proceedings of ICANN-99, 9th International Conference on Artificial Neural Networks, Institution of Electrical Engineers, London, UK.

Wiener, E. D. (1995). A neural network approach to topic spotting in text, Department of Computer Science, University of Colorado at Boulder.

Y. Yang, L., X. (1999). " *A re-examination of text categorization methods*." SIGIR-99.

Yang, Y. and X. Liu (1999). A re-examination of text categorization methods. Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, ACM Press, New York, US.

Yavuz, T. and H. A. G\uvenir (1998). Application of k-nearest neighbor on feature projections classifier to text categorization. Proceedings of ISCIS-98, 13th International Symposium on Computer and Information Sciences, IOS Press, Amsterdam, NL.

Yin, L. and R. Power (2006). Adapting the Naive Bayes Classifier to Rank Procedural Texts. Proceedings of the 28th European Conference on IR Research (ECIR).

Zelikovitz, S. and H. Hirsh (2001). Using LSI for Text Classification in the Presence of Background Text. Proceedings of CIKM-01, 10th ACM International Conference on Information and Knowledge Management, ACM Press, New York, US.

Zelikovitz, S. and H. Hirsh (2002). Integrating Background Knowledge into Nearest-Neighbor Text Classification. Proceedings of ECCBR-02, 6th European Conference on Case-Based Reasoning, Springer Verlag, Heidelberg, DE.

Zhang, D. and W. S. Lee (2006). Extracting key-substring-group features for text classification. Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining.

Zhang, J., R. Jin, et al. (2003). Modified Logistic Regression: An Approximation to SVM and Its Applications in Large-Scale Text Categorization. Proceedings of ICML-03, 20th International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, US.