Abstract

Information retrieval systems provide access to collections of thousands, or millions, of documents, and by providing an appropriate description, users can recover any document from a collection. Refinement of descriptions to correctly match users' needs is often an iterative process, and retrieval systems can utilise user feedback on selected documents to indicate the accuracy of the description at any stage. The style of description required from the user, and the way it is used to search the document database, are consequences of the indexing method used. The index may take different forms, from storing keywords with links to individual documents, to clustering documents under related topics.

The characteristics of the data and processes involved in information retrieval allow much of the work to be automated. Some processes, such as query refinement and document indexing, are usually accomplished by a computerised system, although document classification and index term selection are more often performed manually. Human involvement in development and maintenance of a document database is time consuming, and prone to error. Intelligent algorithms that "mine" documents for suitable indexing information, and model user interests to aid in query production, reduce the human workload and ensure consistent behaviour.

This report presents an investigation of the use of intelligent algorithms, springing primarily from the field of machine learning, in all aspects of the information retrieval process. A literature survey introduces important concepts of the technology, and describes fielded applications. An experimental comparison of different text classification techniques is also presented.

Contents

Figures

Tables

1. Introduction

The aim of this project is to investigate applications of machine learning technology in the field of information retrieval. Machine learning is a dynamic and growing area of computer science, and its techniques are now being applied in other domains. The most common application has been in expert systems, where machine learning algorithms have taken the place of domain experts for generating classification rules and explanations. However, machine learning programs have been applied to many other classification tasks.

Information retrieval (IR) has many characteristics that are suitable for machine learning. Machine learning uses examples, attributes and values, and information retrieval abounds with these. Most IR systems have thousands of documents (examples), and the plethora of natural language features provide a wealth of attributes. It will become apparent that there are *too many* features and examples in most IR systems for machine learning schemes to process satisfactorily. In addition to the raw data that is available, some information retrieval processes are suited to machine learning. Some of these, such as document categorisation, are usually done manually, and most are classification tasks of some form.

The remainder of this chapter summarises the fields of information retrieval and machine learning. The discussion of information retrieval introduces the different phases of an interaction with an IR system, and issues related to each phase. The summary of machine learning introduces the major paradigms in the field, and shows how they can be applied to information retrieval.

Chapter 2 presents some experimental approaches to intelligent document retrieval, and examines applications of machine learning techniques to different phases of the IR process. Chapter 3 presents applications of machine learning to the automation of text categorisation—a process that is expensive and time consuming when performed manually. The next chapter examines techniques for using user feedback to refine queries. These methods create user models that help a system interpret users' queries and gauge their interests. Chapter 5 discusses an experiment comparing a number of different algorithms on a text categorisation task. Machine learning algorithms are compared with standard IR techniques and text models. The final chapter presents conclusions of the project.

1.1 Information Retrieval

Information storage and retrieval systems make large volumes of text accessible to people with information needs. A person approaches an IR system with an idea of the information they require and the system attempts to fulfil that need. The person (hereafter referred to as the *user* of the system) provides an outline of their requirements; perhaps a list of keywords relating to the topic in question, or even an example document. The system then searches its database for documents that are related to the user's query, and presents those that are most relevant.

The information retrieval process can be divided into four distinct phases: *indexing*, *querying*, *comparison*, and *feedback* (Lewis, 1991). The first refers to the way documents are managed in the database. To make searching more efficient, an IR system stores an abstract representation of the documents in its database, using it to isolate those documents relevant to the users' needs. Most fielded IR systems employ a Boolean vector representation for documents. A set of keywords is stored, along with links to the documents that each word appears in. This method of storing indexing information is called an "inverted file". Standard Boolean operators (AND, OR, NOT) are used to combine keywords forming a query. Suppose a user wishes to retrieve documents about motor racing, but they are not interested in truck racing. A query like *((motor AND racing) OR motorsport) AND NOT (truck)* might be appropriate in this case. The system would look up the words *motor*, *racing*, *motorsport*, and *truck* in the inverted file, and retrieve all documents that are indicated by the first three words and not also indicated by the fourth.

An extension of the Boolean model utilises information about the distribution of words in each document, and throughout the document database as a whole. Using this information, an IR system is able to rank the documents it retrieves and present the most interesting ones to the user first—a major advantage over the standard Boolean approach where ordering of retrieved documents is not possible. From a list of documents ordered by predicted relevance, a user can select several from the top as most relevant to the query. In a system using the Boolean model, users would have to search through all the documents retrieved, and determine each ones importance manually.

The form and descriptive power of a query arise from the method of indexing employed by the system. A query makes use of the indices and the operations provided by the conceptual model used to construct the system. In most cases the user is required to have some knowledge of this conceptual model in order to structure queries correctly and efficaciously. There is a trade-off made between the indexing power of the system's model and the complexity of the queries required to search it. Experienced users of a system with limited scope, such as a database of technical reports, may be comfortable

using a complex query interface. They will know the structure of the index and have a feeling for the specific information that characterises the documents they require. In situations where users vary in ability and experience, such as a public library, queries are usually of a simple Boolean form. Users are only required to enter words indicating their area of interest. However, they may not know what indexing terms are used to describe the documents they want. In a library-like environment the keywords describing the contents of a book are chosen by the people who maintain the library's online catalogue. To locate a particular book, the keywords in a query must be identical to those in the book's catalogue entry. This style of querying requires imagination and persistence on the part of the user, who must be prepared to interact with the system for some time.

Systems that employ simple query methods usually provide other facilities to aid users in their search. For example, a thesaurus can match words in a query with those in the index that have similar meaning. However, the most effective tool exploited by IR systems is user feedback. Instead of having users re-engineer their queries in an ad hoc fashion, a system that measures their interest in retrieved documents can alter the queries for them. By weighting terms in the query according to their performance at selecting appropriate documents, the system can iteratively refine a query until it specifies the user's needs precisely—the user is required only to indicate the relevance of each document presented.

Finally, an IR system's conceptual model defines how documents in its database are compared. In a system using an inverted file of keywords two documents may be considered "similar" if they if they have a certain number of keywords in common. A system that uses a term frequency vector model would consider two documents to be similar when their term vectors are close together in the vector space.

1.2 Machine Learning

Machine learning algorithms attempt to take the place of domain experts in knowledge engineering situations. Automation of IR processes, such as document classification and user modeling, using learning algorithms reduces the workload of information workers, and removes inconsistency introduced by human error.

Langley and Simon (1995) identify five major paradigms in machine learning research, four of which have been applied in information retrieval studies. The paradigms are analytic learning, genetic algorithms, rule induction, instance-based learning, and neural networks. The last four are similar in that they learn from information with very simple structure—examples of the concept being learned are often described by lists of symbolic or numeric attributes. Simple heuristics are applied to generate structures that represent relationships implicit in the data. These four paradigms are used in intelligent IR

systems where the examples (documents) are described by simple features such as word frequency measures.

Analytic learning systems are typically employed to learn proofs or explanations for example situations using background knowledge. By compiling groups of explanations a system is able to reduce the amount of search required to solve similar problems in the future. The background knowledge and complex structure used to store explanations make analytic learning systems unfeasible for information retrieval. These algorithms learn much from expert explanations and require few examples—the antithesis of intelligent information retrieval.

Genetic algorithms are the least frequently applied of the other four paradigms. As the name suggests, they mimic the behaviour of biological genetic systems. Examples are represented as a string of values similar to genes in a chromosome. A population of examples is stored, and at each iteration operators such as *mutation* and *crossover* are applied. Mutation changes some of the values in an example randomly, whereas crossover combines different values from pairs of examples into a new instance. The population of examples is prevented from growing indefinitely by retaining only the "strongest" examples as determined by some fitness metric. Evolution is terminated when the user is satisfied with the strength of the remaining example or examples. In information retrieval the values in each example might represent the presence or absence of words in documents. Thus, each example is equivalent to a vector of binary terms. The evolutionary process is halted when an example emerges that is representative of the class of documents being classified.

Why genetic algorithms have been ignored by the majority of researchers is unclear. The random nature of the genetic operators, and the resulting non-deterministic behaviour of the algorithms, may be a reason. Also, setting control parameters, such as the probabilities of crossover and mutation, to ensure good performance can be difficult.

Rule and decision tree induction schemes are among the most studied and developed of machine learning techniques. The schemes generate an explicit description of the concept represented by the data. As with all learning algorithms, the accuracy and appropriateness of the concept description reflects the quality of the data supplied. The algorithms have only the information present in the data to learn from, and will pick the most potent regularities to create the concept description. If strong patterns appear by chance, or the data is irrelevant to the classification task, the concept description will be inadequate. Algorithms use different techniques to determine which patterns in the data are appropriate for incorporating in the concept description, but they can be generally classed as either *covering* or *divide-and-conquer* algorithms.

A covering algorithm usually creates a set of rules for each class in a concept. Each rule covers many examples of the class in question (*positive* examples), yet selects few examples of other classes (*negative* examples). Terms are added to a rule until it covers only positive examples, which are removed from the data set, and the process is repeated until every positive example is covered by at least one of the rules. The "best" term to add in each case is determined by a heuristic measure during a search of the space of possible terms.

Divide-and-conquer algorithms recursively split the data set until the remaining subsets contain examples of a single class. The concept description created is often in the form of a decision tree, with the ultimate subsets present at the leaf nodes. This is a fundamentally different approach to that of covering algorithms. Although the goal of both approaches is to produce a concise, accurate concept description, typical divide-and-conquer algorithms try to achieve accuracy by creating the smallest possible concept description. Covering algorithms attack the problem from the other direction—they try to produce a small concept description by using the most accurate terms at each step. There is no evidence to indicate that either approach is inherently superior to the other. However, a divide-and-conquer algorithm, C4.5 (Quinlan, 1993), has become the benchmark for inductive machine learning.

Instance-based learning algorithms do not create an explicit concept description that can be used to classify new examples. Instead, training examples are stored, and new examples are compared with these for classification. The representation of stored examples and the mechanisms for comparison differ between algorithms, but implementors of IR applications usually favour simple methods such as the *nearest-neighbour* technique. Here, all training examples are stored verbatim, and form an *n*-dimensional space, where *n* is the number of attributes describing the concept. Examples are compared using a Euclidean distance metric, with all attributes assumed to be of equal importance. The nearest-neighbour approach is most applicable to numeric or ordered data where the distance between two values has some meaning. With symbolic data the distance between any two values is deemed to be 1. Examples compared by a symbolic attribute will be maximally different if the values of that attribute are different.

Instance-based algorithms are, computationally, among the simplest learning schemes, and variations are often considered as models of human learning. Many theories from cognitive psychology are implemented as instance-based learners for evaluation purposes. Extensions to the standard nearest-neighbour algorithms include generalisation of stored examples (Salzberg, 1990) and alternative similarity metrics (Cleary and Trigg, 1995).

Neural networks also lack an explicit concept description. They represent knowledge as weighted links between nodes in a multilayer network, with activation spreading from input to output nodes during the classification process. Learning involves altering the weights on the links until the output nodes give the correct response to the inputs. As the name suggests, neural networks are intended to model the physical aspects of the human brain, and in doing so, model the learning processes also. However, typical neural networks are minuscule in comparison to the brain, and the concepts they learn are correspondingly simple.

All of the approaches to machine learning described here have been successfully applied to real-world problems—the choice of method in each case seems to be based largely on the experience and preference of the researchers involved. This success is also apparent in information retrieval applications, with the paradigms being utilised in similar proportions to other domains.

2.

ML for Intelligent Document Retrieval

Information retrieval has been an automated process for many years (Frakes, 1992), but only since the late 1980's have researchers applied machine learning technology in this area. The IR process can be divided into four distinct phases: indexing, query formulation, comparison, and feedback (Lewis, 1991), all of which provide opportunities for learning. Researchers tend to focus on one of these subprocesses when trying to improve the performance of their IR systems. For example, clustering techniques (Martin, 1995) build links between related documents so that simple indexing techniques become more effective, and user modeling techniques attempt to extract as much information as possible from users' queries (Bhatia *et al.*, 1995; Krulwich, 1995).

The unstructuredness and diversity of natural language text causes problems for machine learning algorithms applied to any area of IR (Seshardi *et al.*, 1995). The sheer number of features that could be used for classification overwhelms most ML schemes, and selecting an auspicious subset of features is a difficult task. This diversity is also problematic when several collections are searched to satisfy a single query. Direct comparison of ranking scores across collections is not valid because each collection's feature set, and the distribution of those features, is unique.

This chapter explores ideas that address the issues outlined above. First, the importance of the language model used to represent text is discussed along with a number of these different representations. Following this, methods for extracting valuable features from the text are presented. Third, the problems of retrieval in multiple-collection systems are discussed, and two learning algorithms designed to overcome them are presented. The final section covers document filtering; a process employed in situations where users' interests remain constant and the document database is continually changing.

2.1 Models of text and text representation

For an IR system to categorise documents effectively it must extract enough information from the text of the documents to discriminate examples of each category. The system must create a model of the documents' text that somehow captures the nature of their content. Natural language has a myriad of properties that can be used to create these models, but most are intangible, and without a reasonably solid working knowledge of the language in question a processor (human or computer) cannot make use of them. In the information retrieval domain speed of processing is an important factor, so only a small set of easily extracted features can be used by any single system.

One of the most popular text models is the "bag of words," or *vector*, model, where a document is represented by a set of *terms* (Robertson and Sparck Jones, 1994; Lewis, 1991; Witten *et al*., 1994). All documents that can be generated from a given set of terms

form an *n*-dimensional space, where *n* is the number of terms. Terms are usually words, and a vector represents the number of occurrences of each term in a document. All other information implicit in the text, such as the meaning and order of words, is lost once the document's vector representation has been computed. However, the simplicity of this model makes it one of the most popular in IR.

Documents are considered similar if their term vectors are close together in the vector space. The standard method for determining the distance between vectors is the *cosine measure*:

$$cosine(Q,Dd) = \frac{Q \ D_d}{|Q||D_d|}$$

where *Q* is the term vector for the query and *Dd* is the term vector for the document *d*. This can be calculated using the formula:

$$cosine(Q,Dd) = \frac{1}{W_q W_d} \sum_{t=1}^{n} w_{q,t} \ w_{d,t}$$

where *Wd* is the Euclidean length of document *d*, and *wd,t* is the weight of term *t* in document *d*. The cosine measure calculates the cosine of the angle between two normalised vectors. A value near 1 indicates that two documents are similar, whereas a value close to 0 indicates two documents are very dissimilar, the angle between them approaching 90. The continuous nature of the function means that documents can be ranked in order of similarity to the query vector.

The loss of word order information causes the vector space to contain many documents that are not examples of natural language, because they violate the language's grammar. Many documents will also map onto the same vector simply because the same words appear the same number of times in both documents. This effect is more pronounced when the dimensionality of the vector space is reduced. Limiting the number of terms is sometimes necessary to make classification of large collections practical. In these cases words may be excluded because they convey little discriminatory meaning (stopwords), occur too infrequently, or are extensions of other words (stemming).

Most intelligent IR systems that use symbolic learning algorithms use the term vector model of text, because it provides a finite set of attributes (Apte *et al.*, 1994a; Apte *et al.*, 1994b; Cohen, 1995). The attributes have very simple structure—either a binary value indicating term appearance (for example, Crawford *et al.*, 1991), or a term frequency measure (Apte *et al.*, 1994a). A set of term vectors representing the contents of an IR system's database is no more than a table of integers.

Goker and McCluskey (1991) use a variation of the term vector technique. Inputs to each learning session are *term structures* specifying the name of a term, the number of

documents it appears in, the number of these documents that are relevant, the term's weight, and where it originated from. This last component will specify either a user's query, or a merging of a query and terms selected from documents earlier judged as relevant. Concepts are defined with *concept term structures* that are composed of the name of a term, an integer indicating its strength, the number of relevant documents it appears in, and a list of all its sources. A concept description is a set of concept term structures.

Witten and Bell (1990) discuss another type of word-frequency model in which the appearance of each word is considered a separate Poisson process. The total number of words in a sample of text along with counts of words that appear once, twice, and so on, are used to predict the distribution of words in a related document. Witten and Bell describe how this technique was used to determine whether a newly discovered poem was written by William Shakespeare. Using the 885000 words of Shakespeare's complete works as the sample text, it was estimated that there would be 4.22 words in the new 430-word poem that Shakespeare had never used before. There were in fact seven novel words, just over 1 standard deviation from the estimate. This is not conclusive proof that the poem was penned by the Bard, but the same technique can be used to rule out other possible authors.

Other models of text originate in the domain of text compression. In this field researchers work with the idea that a more accurate model of a document will allow it to be compressed more effectively. Many compression methods utilise the order that terms appear in the text, and use this information to group them into strings of arbitrary length, forming the *alphabet* of the document. An index identifying each string is used in place of the string in the compressed version of the document. To decode the compressed document, a table of strings is consulted, and indices are replaced with the appropriate piece of text.

The ZEROWORD and FIRSTWORD algorithms (Moffat, 1987; Bell *et al.*, 1990) use the existing groupings of characters in a document as the basis for dividing the text into shorter strings. Sequences of characters delimited by non-letters are used in the alphabet—ideally these map to actual words in the text. To compress a document, known words are replaced by their index, and novel words are specified in full along with their index which is used thereafter. Moffat shows that this type of compression is very effective for types of text files that are readily divisible into words.

ZEROWORD uses a context-free model, whereas FIRSTWORD uses the existence of one word to predict the next. The PPM algorithm (Bell *et al.*, 1990) also uses context information. PPM compresses at the character level, and uses some finite number of preceding characters to create the context for the next. Terms (characters) known in the context of the previous term are in the previous term's *first-order* context; terms known in

the context of the previous-but-one term's context are in that term's *second-order* context, and so on. This process can continue indefinitely and is limited by time and space constraints. However, the more contextual information available for prediction, the better the prediction will be.

Compression algorithms can be used to categorise documents in a very simple manner that invokes the minimum description length principle (Lang, 1994). A sample of text from each category is used as "training" data for the compression algorithm. The document to be classified is added to each training sample in turn and the sample that compresses the document into the fewest bits is judged to have the same category as the document. The relative degrees of compression achieved by each sample can also be used to rank the categories. An intuitive notion is that a better compression algorithm will achieve better classification accuracy, because the algorithm is able to use stronger patterns in the sample text to compress the unlabelled document (Lang, 1994). However, experimental results do not support this (see Chapter 5).

Cohen (1995) also used term order to aid in text categorisation. Using the relational learning algorithm FOIL (Quinlan, 1990), Cohen was able to generate rule sets that utilised relationships between words such as *near* (1, 2, or 3 words apart), *after*, and *successor*. Word occurrence relations, analogous to the term appearance values in the vector model, were also used in the rules. For example the relation *learning(d3,2)* would indicate that the word "learning" appeared in document *d3* as the second word. The non-word relations such as *near*, provide background knowledge on word positions. For example, the definition *successor(4,5)* indicates that the fifth word of a document immediately follows the fourth.

2.2 Feature extraction

The text model used by an IR system defines the *features,* or types of information, that it must extract from the text. It is these features that the learning component of the system uses to find patterns in the text from which it can create classifiers. The better the system is at selecting auspicious features, the better its chances of generating an accurate concept description. Several researchers (Holte, 1993; Lewis and Ringuette, 1994; Krulwich, 1995) have asserted that feature extraction is the most critical stage of the learning process in any domain.

Problems in feature extraction arise for a number of reasons (Martin, 1995; Lewis and Croft, 1990). First, the large number of synonyms in English allow a similar idea in two documents to be expressed with different words. Related documents would have to use enough of the same words for the similarity to be recognised. Also, many words will appear in both documents simply because they are in the same language. These words

have no discriminatory power, and will be evenly distributed throughout most documents. Sometimes, the same words will appear in two documents but have different meanings. Two documents may even be considered similar because they both omit a number of important words. This is often the case in two-class categorisation situations where the documents in the negative class are similar only in that they are not related to documents in the positive class. In clustering situations, however, it is necessary to further differentiate these documents.

Standard IR solutions to these problems include the use of a thesaurus to match synonyms (Martin, 1995) and *stoplists* to identify words that are too common (Apte *et al.*, 1994a; Witten *et al.*, 1994). Dealing with words that do not exist in documents is not so straight forward. Usually the situation is ignored, but Martin (1995) treats two such cases differently in his text clustering system. When comparing documents, the absence of a keyword in both of the term vectors indicates the documents *could* both contain the word, but the information is not available. If, however, one of the vectors contains the word and the other does not, the documents are assumed to differ with respect to that word.

Other approaches to extracting the most distinctive parts of documents go beyond using single words as features. The system described by Krulwich (1995) identifies semantically significant *phrases* in documents of a bulletin board-style environment. Phrases can consist of words in the keywords, names, title, and subject fields, and certain patterns in the text fields of documents. For example, a single word that is fully capitalised, or a short phrase of one to five words that is in a different format, such as bullets, diagram labels, or section headings, is usually regarded as significant. Each document is then rechecked for the complete set of phrases, and the system learns a subset of phrases that best describes the category. The learning system uses a simple induction algorithm—the emphasis of the study being to develop more sophisticated feature extraction heuristics. Comparison of a number of different learning algorithms, from neural networks to standard decision tree learners, underscored the point that feature selection is the most important part of the learning process.

Conrad and Utt (1994) use a similar technique to extract names of companies and people from *Wall Street Journal* articles. Company names are recognisable when they include words such as “Corporation” and “Inc,” and personal names are capitalised and may be preceded by a title such as “Ms.” Tables of common personal names, titles, and so on, are stored by the system and used to identify important objects in the articles. The system is being extended to recognise synonyms and abbreviations such as “Digital Equipment Corporation/DEC.” Although these techniques are very simple, the researchers show figures of 89% for precision and 79% for recall on a test database of 139 articles.

Krulwich's system (and to a lesser degree Conrad and Utt's system) makes use of formatting information in the text not only to determine the relative importance of phrases, but also to determine their extent. If a system only has access to the pure text of documents, the problem of determining the appropriate granularity with which to divide the text arises. The standard grain size is the word—these are easily distilled from text, and are obvious distinct units of meaning. However the difficulties inherent in language outlined above have led researchers to examine larger sections of text such as phrases, sentences, paragraphs, and groups of paragraphs.

A number of studies have shown that simple template- and parser-based algorithms, and simple statistical phrase formation algorithms are good at finding important phrases. In a study of clustering syntactic phrases, Lewis and Croft (1990) used a simple syntactic method. They defined a syntactic phrase to be any pair of non-function words that were heads of syntactic structures connected by a grammatical relation. For example, a verb and the first noun of the following noun phrase form a syntactic phrase. Extracted phrases are clustered into small groups containing a seed phrase and the phrases most similar to it. The researchers show that using phrases instead of single words in this context results in slightly improved recall and precision.

Hearst's (1994) TEXTTILING system uses linguistic analysis to determine where subtopic changes occur in long texts. These boundaries are used to divide the text into subsections, and usually occur at paragraph boundaries. Each subsection has its own vocabulary of words relevant to its topic, and the distribution of these words would be skewed towards related sections. A global word-based approach to comparing the documents might miss these relationships, because the distribution of words across the whole document would be relatively sparse. Salton *et al.* (1994) also match sections of full texts to determine similarities between documents.

2.3 Multiple-collection retrieval

Most IR systems have a single text database that is consistent in terms of indexing and content. However, some systems are required to search a number of databases to satisfy a user's query (Towell *et al.*, 1995). For example separate collections might be used for *football* and *motor racing* documents. A query about *world champions* might find relevant documents in both collections. Problems arise because the individual collections will have different distributions of terms. Identical documents in both collections will have different scores for the same query, and comparison of scores between documents from different collections becomes meaningless. These effects are strengthened when the different collections each have their own servers, and use different ranking and retrieval strategies.

A typical query to a multiple-collection system asks for a certain number of documents, *N*, selected from *C* collections, and ranked in order of relevance. The problem of merging the documents in a valid order is called *collection fusion* (Towell *et al.*, 1995). Simple solutions such as selecting the top *N/C* documents from each collection, or assuming the relevance scores are compatible across collections and selecting the overall *N* top scoring documents, perform very poorly. Towell *et al.* introduce a collection fusion scheme, *modeling relevant document distributions* (MRDD), that uses the relative document distributions of training queries to learn the appropriate number of relevant documents from each collection to retrieve for a new query. The algorithm models the distribution of a query by averaging the distributions of its *k* nearest neighbours in the training data. The number of documents to retrieve from each collection is calculated using a maximisation procedure (not described in the article), and the order of presentation is determined by a random process biased toward collections with the most remaining articles.

A second learning scheme described by Towell *et al.* clusters training queries that retrieve many common documents from each collection. The average of the query vectors in each cluster is the system's representation of the topic covered by those queries. Each cluster is given a weighting for each collection proportional to the number of relevant documents retrieved from the collection. After training, new queries are matched with the most similar cluster and that cluster's weighting model is used to determine the proportion of documents to retrieve from each collection.

Both these learning algorithms are significantly better than the simple schemes described above in terms of precision given a set recall figure. Of the two, MRDD performs the closest to retrieval on a single collection—equivalent to combining the collections and using a single server—achieving precision figures within 10% for recall of 10 to 200 documents.

2.4 **Document filtering**

Most IR applications involve a relatively constant set of documents and a heterogenous array of information requirements from users. However, there are situations where the users' interests remain constant and the documents that satisfy those interests are continually changing. The systems model the users' interests (see Section 4.2) to *filter* the continuous stream of information for articles of interest.

Krulwich's system (Krulwich, 1995) filters information from a number of sources in a Lotus Notes environment, using a term vector approach. Documents constitute four groups, ranging from general discussion to memos and bulletins. The number of documents relevant to the average user is estimated at less than a dozen of the 3000 that appear daily. By presenting predicted relevant documents to the user, and obtaining information on which of these are relevant and why, the system is able to fine tune each user's filter.

NEWSWEEDER (Lang, 1995) also uses the token vector representation for each article. The system also presents to the user documents it predicts are relevant. Users indicate on a scale their actual interest in each document, and NEWSWEEDER analyses the contents of each document generating a linear regression model for predicting the users' ratings in the next iteration.

3.

ML for Text Categorisation

Text categorisation is the process of automatically classifying documents into one of a number of categories. Classification of documents is required at several stages of the information retrieval process. A user's queries must be classified to determine which documents in the collection are relevant. A query may be treated as a document in its own right, or as a description of the desired concept. The category represented by the query may not be represented explicitly in the database—it is the system's indexing mechanism that determines how documents fitting the category description are located.

Documents also require labelling in applications such as library cataloguing systems and newswire stores. Intelligent systems can predict the category of new documents by learning a concept description from a set of training examples. This is a standard ML task, although the high dimensionality of the data involved requires special consideration.

Automation of the labelling process has a number of advantages. The volume of work involved in most text classification tasks is enormous, and performing this manually is time consuming and expensive—once an intelligent system has achieved a satisfactory level of accuracy it can perform the task in a fraction of the time. Section 3.2 describes a method of selecting training examples from a set of unlabelled documents that tries to minimise the number a supervisor must initially label by hand. The technique is shown to give a 500-fold reduction in the number of documents requiring manual classification.

Another benefit is consistency of classification. Human experts can disagree on classification of documents, and even a single person may classify documents inconsistently (Apte *et al.*, 1994a). Differences in background knowledge and ability are responsible for inconsistent human classification.

This inconsistency also creates a problem for automatic classification systems. Human performance is regarded as ideal in classification tasks and intelligent systems must try to perform to this level. Thus the optimum threshold for classification accuracy of 100% is almost certainly unattainable (Apte *et al.*, 1994a). All documents used to train a classification system must first be labelled by a human expert. A system attaining 100% accuracy on this training data will have incorporated the expert's errors and inconsistencies into its category descriptions. Had another person classified the documents, the concepts may have been described differently.

Many ML techniques have been applied to text categorisation problems, but because of the uncertainty about absolute measures of performance no great success has been claimed. The next section describes algorithms that have been used and the concept descriptions they create. It should be noted that none of the systems discussed are considered *ineffective* by their developers.

3.1 Algorithms, concept descriptions and data structures

Development of learning systems in IR focuses on either of two aspects: applying an existing "off the shelf" algorithm in some manner, or developing a new learning algorithm specifically for an IR application. The choice of approach depends on the expected application of the system, and the experience of the developers.

Standard algorithms

Many of the "standard" ML algorithms have been used in intelligent IR systems. In general, the more simple algorithms like ID3 (Quinlan, 1986) are used for text classification problems, such as identifying text features that catalyse learning (Krulwich, 1995; Lehnert *et al.*, 1995; Soderland and Lehnert, 1995). Chen (1995) used ID3 as a control for an experiment with ID5R (Utgoff, 1989), an incremental extension of ID3. The more elaborate schemes, such as ID5R, C4.5 and FOIL (Quinlan, 1990), are usually relegated to feasibility experiments (for example, Cunningham and Summers, 1995; Chen, 1995; Cohen, 1995).

One of the defining features of an ML algorithm is the form it uses to represent the description of learned concepts. In IR this concept description is used to classify documents, or explain the existing labels of documents. Inductive learning schemes, such as INDUCT (Gaines, 1991), ID3, and FOIL, are useful for these tasks because they produce simple, easy to understand rules or decision trees (Lewis and Ringuette, 1994). A supervisor can evaluate and augment these structures for classification purposes, or use them to gain insight in human-categorised databases.

Production rules are the most common form for concept descriptions. These may be propositional rules generated by INDUCT (Cunningham and Summers, 1995), or SWAP-1 (Apte *et al*., 1994a), or refined from decision trees (Crawford *et al*., 1991; Cunningham and Summers, 1995); or relational rules generated by FOIL taking word position information into account (Cohen, 1995). Propositional rules generated from text tend to look like those in Figure 3.1. These rules were synthesised from a decision created by CART (Breiman *et al*., 1984). The rules indicate the presence or absence of a particular

if article contains word "bomb"
    then if article contains words "injure" or "kill"
        then **terrorism** article
        else **~terrorism** article
    else if article contains word "kidnapping"

Figure 3.1 Rules representing a decision tree generated by CART from 730 Reuters news articles. The rules predict whether or not articles are about terrorism.

word in a document, although some algorithms use rules containing numerical thresholds; if the occurrences of word indicated by such a rule exceed the specified value the term is true for that document.

Relational terms add an extra dimension to a rule set. Cohen (1995) defined a number of positional relations that could be used in addition to appearance information for each word, allowing rules to specify relationships *between* words in a document. Figure 3.2 shows such a set of rules. Instead of just appearing in the same document, the words "decision" and "tree" in the second rule of Figure 3.2 must also satisfy the *successor* relation—that is, "tree" must directly follow "decision" at some point in the document.

When comprehension of the concept description is not important outside the system, other methods are often applied. Versions of the nearest neighbour technique are used when clusters of documents or queries are generated and compared (Towell *et al.*, 1995).

```
machine_learning_article(S)  learning(S,P).
machine_learning_article(S)  decision(S,P), tree(S,Q), successor(P,Q).
```

Figure 3.2 Relational rules generated by FOIL to classify machine learning documents.

Distance metrics are usually very simple, such as the sum of the differences of term counts (Salton *et al.*, 1994). *Bayesian* probabilistic algorithms also have no explicit concept description. Lewis and Ringuette (1994) used Bayes' rule to estimate $P(Cj=1|D)$, the probability that a category *Cj* should be assigned to a document given the prior probability of the category occurring, and the conditional probabilities of particular words occurring in a document, given that it belongs in the category. Each category is assigned to its top scoring documents in proportion to the number of times it was assigned on the training data. Lewis and Ringuette compared the precision and recall of their Bayesian algorithm with a decision tree learning algorithm and found little difference between the two. The Bayesian classifier was able to handle ten times the number of features as the decision tree learner, although both algorithms peaked in performance between ten and one hundred features.

Hearst (1993) proposed a *case-based reasoning* (CBR) approach to represent the main topics and subtopics in full text documents. A subtopic structure outline of a document is an abstract representation of the document, and can be used as a case for CBR. Document cases are placed in a case network, positioned according to which cases they resemble. Specifying which features may differ and allow cases to remain similar is accommodated by associating a term vector with each section of a document. Cases are

organised first by their main topic, and then grouped according to which and how many of their section term vectors are similar.

Neural networks and genetic algorithms have also been used for classifying documents (Chen, 1995; Towell *et al.*, 1995). As with the more complex inductive schemes, these experiments have usually been performed as controls or feasibility studies. All feasibility experiments in this area show that ML methods can be applied productively to text classification problems when the domain is constrained to a small feature set and a small number of categories. Although ML schemes perform a certain amount of feature selection in generating a concept description, it has been shown that prefiltering the text to remove ineffectual features is necessary.

Specialised algorithms

A number of learning algorithms have been developed strictly for text classification. Bhuyan and Raghavan (1991) developed a probabilistic scheme that uses user feedback to build document relationship clusters. Three graphs are generated from a series of queries to store information about pairs of documents. If both documents in a pair are relevant to a query weight is added to the corresponding edge in the *POS_POS* graph to reflect the similarity of the documents. If the first document is relevant and the second is not, the edge of the *POS_NEG* graph is similarly updated. The third graph, *NEG_POS*, is revised for the reciprocal case. A new graph *G* is generated by summing the weights of the *POS_NEG* and *NEG_POS* graphs and subtracting the weights of the *POS_POS* graph. The graph *G* indicates the similarity between documents, with lesser weighted edges denoting greater similarity.

In their extension to the OKAPI IR system, Goker and McCluskey (1991) define concepts with a set of *concept term structures*. Initially the concept description is bound to the set of all terms that appear in more than one document, with further highly weighted terms added until all relevant documents are covered. This set is called the *active set*; the remaining terms make up the *passive set*. After each user session new terms are merged into each set and existing terms are swapped between sets as their weights change. The active set is limited to 32 terms by moving the terms with lowest strength back into the passive set. After each query the concept description can be used to rank documents initially given the same weight. A document's new weight is the sum of the words that appear in both the concept description and the document.

Inference networks (Haines and Croft, 1993) rank documents based on the probability that the documents satisfy the user's information need. Figure 3.3 shows the structure of an inference network used for information retrieval. Although inference networks were not specifically developed for IR this structure indicates the information necessary for applying the technique to this domain. The *D* nodes represent documents in the system's database, the *R* nodes describe the contents of the documents—documents with similar content are linked to the same *R* nodes, the *Q* nodes represent queries, and the *I* node is the user's information need. The *D* and *R* nodes are constant for a set of documents reflecting relationships between the documents. The *Q* and *I* nodes are created for each query—in this case *(information and retrieval) and (not files).* Probabilities filter down the network from each of the document nodes at the top, and the value that arrives at the *I* node indicates the relevance of each document to the user's information need.

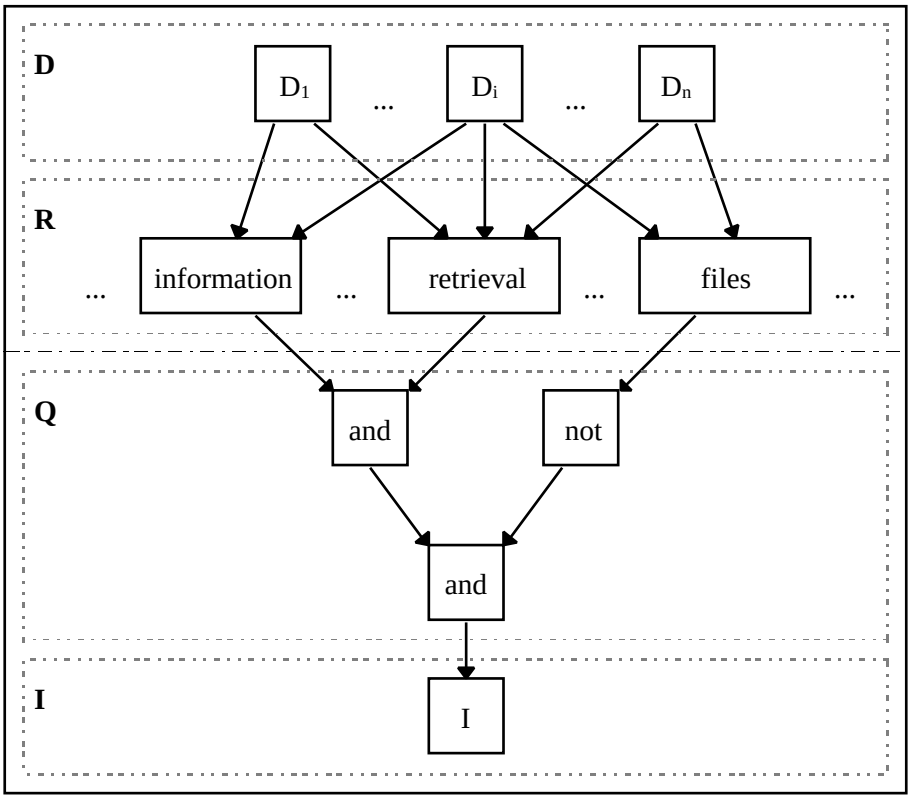

Figure 3.3 An example inference network created from the query *(information and retrieval) and (not files).*

The learning system in NEWSWEEDER (Lang, 1995) is based on the *minimum description length* (MDL) principle. In NEWSWEEDER's domain the MDL trade-off between model complexity and error rate relates to determining how to weight each term's importance, and how to decide which terms to omit for having too little discriminatory power.

The system's term distribution model assumes that the probabilities of the occurrence of any two terms in a document are independent. However, the probabilities are allowed to depend on the length of the document. Distributions of each term are computed for each category, and across all categories. To create a model of each category a choice is made between the category-specific distribution and the overall distribution of each term. If the difference between the probabilities for a term is larger than a threshold the category-specific distribution is used for that term, otherwise the overall distribution is used. The threshold value represents the cost of including parameters that specify the category. Finally, linear regression is used to produce a continuous prediction of the user's ratings for new documents.

3.2 Uncertainty feedback

Because of the vast amounts of text stored in many databases, it is impractical to use it all as training data for an intelligent text classification system. However, any sample selected as a training set should include many documents that are good discriminators for the categories that exist in the collection. Random sampling is usually unsuitable, particularly when the distribution of categories is skewed and the sample size is relatively small. Relevance feedback performs a kind of sampling by selecting documents that are representative of a given category, but as the learning system improves many of these documents provide no new information for the system to learn from.

Lewis and Gale (1995) present an algorithm for sequentially selecting a subset of training documents in a similar manner to relevance feedback. However, *uncertainty feedback* does not provide the learning algorithm's supervisor with documents predicted to belong to the category in question. Instead, it presents documents about which it is most unsure of the appropriate category. By labelling these documents the supervisor creates a training set of example documents that are more disparate than those selected by relevance feedback.

The system starts with a small set of examples, selected either randomly or as being particularly representative of some categories, and a classifier is created for these documents. The system then proceeds through the *classification–feedback–create a new classifier* loop until the supervisor judges the current concept description accurate enough. Lewis and Gale show experimentally that this point can be reached with as much as a 500-fold decrease in the number of examples required by random sampling. Uncertainty sampling is also shown to be more effective than relevance feedback even when the number of examples in each category is small enough that relevance feedback does not select so many redundant examples.

4.

ML for Query Refinement

The user's input to an information retrieval system is the query. In most cases queries are Boolean expressions over document keywords, but may also be whole or partial documents. Some systems accept natural language queries which are usually translated into a Boolean expression for retrieval purposes. However, Boolean queries are not ideal for representing users' information needs (Salton *et al.*, 1994; Hearst, 1994). Inexperienced users often find them difficult to structure effectively, and they provide no facility for indicating the relative importance of individual terms. To overcome these problems many IR systems make the querying process an iterative one. Using the initial query as an outline of the user's information need the system progressively refines the concept until the user is happy with the set of documents retrieved.

This chapter discusses ML techniques applied to three areas of the query refinement process. First, the problems users have with information retrieval are presented. The second issue is *user modeling*, a process that occurs over a series of interactions with the system as it builds a profile of the user's interests. This mechanism is most applicable where a user's query remains relatively constant in a continually changing information environment. *Relevance feedback*, a standard query refinement technique, is the third topic discussed. The section describes how intelligent systems can gain information from the user about the pertinence of selected documents that match the query.

4.1 Users don't know what they want

People use information retrieval systems when they need information they believe exists in documents in the system's collection. Because of the volume of text in the database, and the range of topics covered, the user cannot know exactly how to retrieve all, and only, the documents relevant to their information need—a problem made worse when they have only a vague idea of the information they want. For example, a student researching a topic for a paper will initially have only a few keywords to characterise their information need—they would not have titles and authors of every paper in the collection related to their topic.

The user's unclear idea of the information they want and how to get it means the first query will be very general; perhaps a simple concatenation of the known keywords, or an example document related to the topic. Unfortunately, characteristics of natural language, such as homonymy, synonymy, and ambiguity (Lewis and Ringuette, 1994; Martin, 1995), make it impossible for the retrieval system to present the user with every relevant document given such an imprecise description. The system is unable to predict, from such a tiny sample of pertinent text, the elements important to the user's information needs. Returning all documents related in some way to the initial set of keywords is the simplest

response to the query, but in addition to relevant documents there would be many with very weak links to the area of interest, some that mentioned it in passing, and others that used similar words for different purposes. The user would have to sift through a large number of documents for any that appear interesting. Even when the system is able to rank the documents it retrieves, the query's lack of discriminatory power causes weight blocks (Goker and McCluskey, 1991), where many documents will have same rank.

An intelligent information retrieval system must help the user by not overwhelming them with copious amounts of text. It should aid the user in refining their query, so that they are eventually afforded a smaller number of documents more illustrative of their information need. The system needs to extract from the user enough information to determine which of the documents related to the initial query are actually important. This is necessarily an iterative process, as the system attempts to converge on a much more explicit description of the user's information requirements.

4.2 User modeling

User modeling is a process that occurs over a series of interactions with the system as it builds a profile of each user. The technique is most often used where a user's requirements remain constant. In these situations it is the user's *interests* that the system tries to identify, although other information about the user can also aid an IR system. Knowledge of the user's background and experience can be used to interpret queries individually and help in document comparison.

Modeling user interests

In many information retrieval domains a user's interactions with the IR system occur over a period of time. Each interaction is a discrete event unrelated to any of the previous interactions, whereas other domains involve continuous interaction involving the same topic. For example, a search for books about functional programming languages might be a one-off exercise with a library's computerised cataloguing system, whereas following a thread in a USENET newsgroup would be an ongoing procedure, with interactions occurring once or twice a day. In the former case the user is usually willing to endure an extended dialogue with the system in order to get the material they want. The system will supply a number of books on the subject in question and the user may not need to return for some time. In the latter case, however, the user would not want to have to search the entire USENET feed to find a handful of articles that are usually of passing interest or short-lived pertinence. Often the effort required to find these articles in a system with little structure is more than the user is willing to spare. In this situation the user wants a means of keeping track of interesting articles, and separating them from the mass of uninteresting text.

Most news reading systems keep a hotlist of newsgroups interesting to the user who must rummage the groups to determine which articles are of interest, usually on the strength of the articles' titles or a simple keyword search. As the number of groups, and the range of topics, increases, the user's task becomes overwhelming. A trade-off is necessary between looking through fewer articles over their range of interests, and finding postings that are truly interesting (Lang, 1995).

To lessen the strain on the user, intelligent systems try to build a profile of the user's interests during interactive sessions and present articles that match that profile. This *user model* may be generated collaboratively with the user (Lang, 1995; Krulwich, 1995) or from logs of user sessions (Goker and McCluskey, 1991), and is continually updated reflecting changes in the user's interests and the content of the text database.

Krulwich's (1995) system searches for new articles in a Lotus Notes environment that match a user's profile, updating it as the user indicates which of the system-selected articles are of interest. After the nightly search of the database the system has collected a number of articles that match the current user profile. These documents are presented to the user who scans them for items of interest. As articles are selected the user is asked to indicate why they are interesting, be it the subject area, author, or whatever. These "reasons" become categories for the system to learn.

Lang (1995) describes a similar system, NEWSWEEDER, that learns user models for identifying interesting USENET news articles. In addition to providing the services of a traditional newsreader, NEWSWEEDER generates virtual newsgroups tailored to individual users' preferences. Each user's personalised list is ranked according to a predicted interest rating for each article. The user selects any articles from the list that appear interesting, reads them, and rates them from 1 (essential) to 5 (never want to see anything like it again). All other articles are marked "skip" indicating that the user does not even want to read them. Ratings are collected, and overnight the system learns a new model that is used to predict the ratings the user will give to the next set of articles appearing in their personalised newsgroup.

The OKAPI IR system (Goker and McCluskey, 1991) provides retrieval services for three databases at City University in London. OKAPI ranks retrieved documents by the probability that they are relevant a the user's query. This probability is calculated using a modified version of the term weighting formula in Robertson and Sparck Jones (1994). The system also uses standard relevance feedback techniques (described in the next section) to expand the user's initial query.

Goker and McCluskey describe an incremental learning algorithm that, when used with OKAPI, forms a model of users' areas of interest. The algorithm learns from logs of user sessions that contain details of search terms, search and response timings, number of

references examined, and other information. The concept descriptions formed can be used next time a user carries out a search on a similar subject, with emphasis on alleviating weight block problems.

The ideas discussed for these three systems can also be applied to *user group models*. Instead of building a profile of a single person, the system models a number of users, and uses the information to select documents relevant to individuals, and the group as a whole.

Modeling user concepts

Bhatia *et al.* (1995) introduce a different kind of model that describes concepts in the user's vocabulary, and can be used to interpret and refine queries in a manner appropriate for the individual user. First, the system "interviews" the user to determine objects, and relationships between these objects, important in the user's universe of discourse. The interviewing process is based on *personal construct theory* (Kelly, 1963). Objects named by the user (called *entities* in personal construct theory) are stored in a table, and presented back to the user in randomly selected triples. The user must then provide the system with a perceived bipolar property of the entities (called a *construct*) that can be used to differentiate one of the entities from the other two. The construct becomes a row of the table, and the user is asked to enter a value from a predetermined scale that indicates the relevance of the construct to each entity. The user may spontaneously add new entities and constructs throughout the interview, in addition to normal interaction with the system.

The constructs supplied by the user represent their personal *point of view* of the objects around them based on their environment and background (Kelly, 1963). A different person may categorise similar objects with entirely different constructs. For example, one person may not find the construct *black-vs.-white* relevant to the entity *time of day*, whereas a person who regularly uses a train timetable that divides the day into strips of black-on-white and white-on-black text may regard this as a relevant discriminator. *Black-vs.-white* is not a natural property of *time of day*, it is one devised by the second person as a distinguishing feature of the entity. Constructs provide the basis on which objects are understood, and may not necessarily reflect their actual differential groupings. Therefore, the system will develop a different structure for each user's world view.

The user profile is used by the system to map concepts important to the user to keywords occurring in documents in the database. The user selects a small set of documents with which they are quite familiar, with at least one document containing an example of each concept to be used during querying. The system then creates a mapping for each concept to a set of index terms obtained from the appropriate example documents. Documents are represented in term–vector format, with terms weighted using

the *tf-idf* method. Terms with weights above an empirically determined threshold are selected as representative of the document. The user can then enter queries using their own vocabulary and concepts, and the system will map these to appropriate production rules and keywords to search the remainder of the database.

4.3 Relevance feedback

A user's initial query to an IR system will usually retrieve only a few interesting documents mixed in with any number of uninteresting documents that weakly match the query. This is due to the general nature of the query, which often has only a few terms, and the system's imperfect representation of the contents of its documents (Lewis, 1991). In a probabilistic system the documents will be ranked in some manner, but the user is still required to scan through the list to determine where the cutoff point for irrelevant documents lies. Often the user will be happy with a short, incomplete list of documents, or a single reference, and the results of their initial query will suffice. However, when a more complete list of relevant documents is required, modification of the initial query is necessary. Users may do this themselves, but they have usually stated their requirements to the best of their ability in the initial query, and will be unsure how to make effective refinements.

The *relevance feedback* technique (Robertson and Sparck Jones, 1994; Harman, 1992; Lewis, 1991) expands queries by first extracting significant parts of documents retrieved by the user's query and asking the user to indicate each part's relevance to their requirements. The user may be presented with whole documents, abstracts, selected passages (Allan, 1995), keywords, or other terms the system deems representative of the results of the initial query. These items are usually ranked, and their number limited to reduce the risk of including worthless terms. The selected terms are then added to the initial query, existing terms reweighted according to their performance in the previous search, and the query processed again. This procedure is repeated until the user is satisfied with the list returned by the system.

A learning system can use terms or documents indicated as relevant as positive examples, and those deemed not relevant as negative examples (Lewis, 1991). The user's first query becomes the initial concept description, and the system attempts to refine this as the relevance feedback process continues. The small size of the initial query limits the number of features to learn from, inadvertently combating the dimensionality problem. The feature set is expanded as new terms are found to be important to relevant documents, but it will never approach the number of terms present in the documents. The terms of the initial query can also be given more emphasis than positive examples introduced later, so that they always have more weight during the learning process.

The amount of training data available to the learning system via relevance feedback is relatively small with a bias towards positive examples. This bias occurs because the system is trying to find positive examples of the user's desired concept, and presents the user with items that best match the concept so far developed. Lewis (1991) suggests this bias is probably good in this situation.

Haines and Croft (1993) describe extensions to an inference network model of information retrieval to include relevance feedback techniques. The researchers investigated a number of variables pertaining to relevance feedback, such as term reweighting, additional terms, and new term weighting methods.

In the inference network model queries are represented as links between the query nodes (*Q* nodes of Figure 3.3), and the information need node (*I* node of Figure 3.3). Query term weights are determined by the *weighted sum* form of the link matrix at the *I* node. To add terms determined by relevance feedback new links are added between the *I* node and the new *Q* nodes created for each new query concept. The link matrix weights are re-estimated using the sample of relevant documents. The weight associated with a query term is used to predict the probability that the information need is satisfied given that a document has that term. Relevance feedback involves re-estimation of this probability.

Bhuyan and Raghavan (1991) store relevance feedback information from a number of users in three graphs denoting different relationships between pairs of documents. When enough information has been stored the graphs are combined and used to form clusters. The system tries to obtain a consistent classification for documents over a number of user interactions. Documents considered jointly relevant by a number of users are clustered in the same group, and the clusters used in later searches to retrieve related documents more efficiently.

The intelligent IR systems of Goker and McCluskey (1991), Lang (1995), and Krulwich (1995) all use relevance feedback as part of the learning cycle. The technique is used to inform the system which documents or terms are useful as discriminators of documents the user is interested in. These terms are used as features for the learning algorithm to work with.

Chen (1995) used relevance feedback in an iterative text categorisation scheme using the ID5R version of Quinlan's ID3 algorithm. ID5R is an iterative decision tree learner that rearranges the structure of the tree to accommodate new examples as they are presented. Chen initially presented the algorithm with a set of keyword groups representing positive examples of the desired concept—a similar set of negative examples was also supplied, as ID5R requires both positive and negative examples to learn a concept description. The decision tree resulting from learning on these examples was then used to

search the remainder of the database, and the set of new documents retrieved was presented to the user for the relevance feedback step. The user classified the documents as either positive or negative examples, and these were then used to update the decision tree. This process was repeated until the entire database had been correctly classified by the decision tree.

5.

An Experimental Study of Text Categorisation

The purpose of this experiment was to compare the performance of two standard machine learning schemes with a number information retrieval algorithms and text models. Lang (1994) performed a similar experiment in the initial development stages of his NEWSWEEDER system. Lang showed that the *term-frequency–inverse-document-frequency* algorithm achieved the highest accuracy in classifying USENET news articles from twenty newsgroups. The algorithm was compared with methods such as a single *default rule*, *random guess*, *neural networks*, and a *compression-based* technique. The number of articles used for training data was varied from 1000 to 20000, with all schemes showing improved performance on larger data sets. The *random* and *default rule* methods were not applied in practice, and an expected value of 5% was used as the worst case threshold for the other methods. The compression-based method employed a simple MDL principle. Each training newsgroup was compressed using the UNIX GZIP utility, and each test article was classified as belonging to the newsgroup which compressed it into the fewest bits.

5.1 Method

Training articles were selected randomly from ten newsgroups, and concatenated into a single file for each group. The newsgroups were selected so that there was little chance of articles being crossposted amongst them. This resulted in a diverse set of topics, but the purpose of the experiment was to gauge the relative capabilities of the schemes, rather than their ability at finegrained discrimination.

Enough articles were taken to give each training file approximately 35000 words, except for *rec.arts.books.tolkien* which could only manage 18000 words before it was exhausted for the day. There were approximately 1600 training articles in total, all of which had their headers removed, along with punctuation, numbers, and other symbols. All letters were converted to lowercase. The only other symbols allowed in words were apostrophes and hyphens, in appropriate places for standard English. Summaries of the newsgroups, along with an example article, are given in the appendix. One hundred test articles, ten from each of the newsgroups, were selected randomly two weeks after the training articles were obtained.

	First run—unstemmed	Second run—stemmed
Average words per article	219	221
Maximum	3577	3628
Minimum	16	15
Standard deviation	391	396

Table 5.1 Summary statistics for runs on unstemmed and stemmed news articles.

Articles were classified in two runs: first, with the words of the article only; second, with the words stemmed using the PORTER stemming program (from the source code in Frakes and Baeza-Yates (1992)). Table 5.1 gives summary statistics for each run.

The schemes used were COMPRESS, GZIP, PPMC, ZEROWORD, FIRSTWORD, WORDAP, TFIDF, C4.5, and INDUCT. To provide an optimal performance threshold the 100 test articles were also classified by thirteen human participants. A summary of each algorithm is given in the appendix.

COMPRESS, GZIP, and PPMC are all compression programs, and were used to test the hypothesis that better compression algorithms will achieve higher classification accuracies. ZEROWORD and FIRSTWORD are also compression algorithms, but they use word-based techniques, unlike the first three programs which are character-based. These two algorithms are intended for compression of text that is readily divisible into words, such as natural language and computer program source code. Prediction using the compression programs is based on the amount of compression achieved when the test article is concatenated to the training file. The predicted class is the newsgroup that compresses the test article into the fewest bits.

The WORDAP algorithm uses the Poisson process model of word appearance, and classifies the test article with newsgroup that most accurately predicts the number of novel words in the article. TFIDF is an implementation of the term frequency/inverse document frequency algorithm using the cosine measure of similarity.

C4.5 and INDUCT are standard ML algorithms, and very flexible in terms of training data and concept descriptions. The algorithms used data in frequency and binary appearance format to generate two different types of concept description. The first created a definition of each newsgroup separately. Test examples were classified by all ten concept descriptions—a correct prediction arising when all ten classified the article correctly. The second type was a single definition that classified all ten classes.

To lessen the effects of the "curse of dimensionality" two methods were used to reduce the number of terms in the data used by the ML schemes. The first method is adapted from Apte *et al*. (1994a)—the 150 most frequent words in each training file were selected, and stopwords were removed, leaving approximately 80 terms for each newsgroup. These words were combined in the 10-class task giving approximately 350 terms. The second method used 1R (Holte, 1993) to select between 80 and 400 terms for each training set—the actual number depended on the accuracy cutoff point which was set to give at least 60 terms. In some cases this point covered several hundred words.

The human participants were asked to classify the unstemmed test articles only, and were given the names of the newsgroups, rather than the 1600 articles, as "training data".

Articles were correctly classified by a scheme if the correct newsgroup was ranked first.

5.2 **Results**

	Unstemmed		Stemmed	
Scheme	Correct–%	Compression–bpc	Correct–%	Compression–bpc
Human–average	89	N/A	Not performed	N/A
TFIDF	84	N/A	85	N/A
PPMC	64	2.41	65	2.42
GZIP	47	2.34	52	2.34
COMPRESS	23	3.28	26	3.28
ZEROWORD	52	2.66	45	2.49
FIRSTWORD	48	2.28	41	2.12
WORDAP	61	N/A	65	N/A
C4.5/2class/b/1R	25	N/A	Not performed	N/A
C4.5/2class/f/1R	17	N/A	Not performed	N/A
C4.5/2class/b/f-s	52	N/A	49	N/A
C4.5/2class/f/f-s	47	N/A	50	N/A
C4.5/10class/b/1R	68	N/A	Not performed	N/A
C4.5/10class/f/1R	63	N/A	Not performed	N/A
C4.5/10class/b/f-s	61	N/A	67	N/A
C4.5/10class/f/f-s	65	N/A	66	N/A
INDUCT/b/f-s	<61	N/A	Not performed	N/A
INDUCT/f/f-s	<64	N/A	Not performed	N/A

Table 5.2 Results on unstemmed and stemmed news articles.

Table 5.2 shows the results of the runs on unstemmed and stemmed text. The second and fourth columns show the number of test articles correctly classified by each scheme. The third and fifth columns show the rate of compression achieved by the applicable schemes. The eight C4.5 schemes specify the different combinations of concept description—*10class* for the ten-class decision tree, and *2class* for the ten two-class decision trees; data type—*b* for binary and *f* for frequency; and term selection—*f-s* for frequency selection and *1R* for 1R selection. For example, "c4.5/2class/f/f-s" indicates word frequency data, and most-frequent-word term selection, with ten decision trees.

Table 5.3 shows a confusion matrix of classifications made by the thirteen human participants. The letters correspond to the newsgroups as presented in the appendix. The diagonal from top-left to bottom-right shows correct classifications; all other grid

	A	B	C	D	E	F	G	H	I	J
A	123	3		3		1				
B	2	108	2	11		3		2	2	
C	2	1	115		1	7	1	2		1
D	7	5		109	4	2	2	1		
E		3	9		113			3	2	
F		6				124				
G	1	1					121	7		
H		1					4	117	1	7
I		1	1	2		1			125	
J		20	1	1			1		1	106

Table 5.3 Confusion matrix showing distribution of news article classification by human participants.

locations indicate misclassifications. For example, the “20” in the tenth row of the second column shows that articles from group B were misclassified as group J on twenty occasions. Human classification accuracy ranged from 82% to 95%, with a standard deviation of 3.3%.

5.3 Discussion

The standard information retrieval algorithm, TFIDF, was the most accurate classification program, being 20% more accurate than its nearest rival in both experiments. However, this advantage is somewhat reduced by the preprocessing of the training data that is required. A dictionary must be built, and term frequency information gathered for each newsgroup. This took a considerable amount of time as there were several megabytes of text to process. In contrast, the compression-based schemes require no preprocessing whatsoever—each newsgroup is simply concatenated to the test article and piped through the compression program. The difference in size between the resulting file and the original file is all the information required for comparison.

PPMC, WORDAP, and the 10-class C4.5 scheme were similar in terms of accuracy, and at least 10% better than the remainder of the algorithms. PPMC and the other compression algorithms are discussed below; WORDAP was another program that required no preprocessing of the text.

The results for INDUCT are optimistic. It is more accurate to say that the rule sets classified, for example, 100 – 64 = 36 articles *incorrectly*. The DNF rules created by INDUCT can give more than one classification to each test article, and PREVAL (Garner *et al.*, 1995), the program used to evaluate the rule sets, regarded these multiple

classifications as both correct and incorrect if at least one of the rules matching the article had the correct class. Thus, the total number of classifications in the confusion matrix generated by PREVAL exceeded the number of test articles. Articles must be misclassified *only* by a set of rules to considered incorrectly classified, and PREVAL reports these examples by name. Therefore, the value given for INDUCT in the tables above is an upper bound on the true accuracy figure—the actual number of correct classifications in each case would be less than this figure. A lack of time to overcome this characteristic led to the abandonment of INDUCT in the rest of the experiment.

Excluding COMPRESS which performed abysmally in both experiments, the remainder of the schemes all performed with similar accuracy of approximately 50%. The performance of C4.5 is notable in this regard because it was always using fewer than 500 of the more than 20000 words available to the other algorithms.

The remainder of this section discusses the characteristics of human classification, the benefits of stemming the text prior to training, the effects of different data and concept description formats and term selection methods on the ML schemes, and the relative performances of the five compression-based schemes.

Human classification

The human classification result shows the importance of background knowledge in text classification. Many of the participants expressed little interest in the topics of some of the newsgroups—a result, perhaps, of the diverse selection. Yet, even the little knowledge they possessed about these topics was sufficient to allow them to comfortably outperform all but one of the algorithms.

The confusion matrix in Table 5.3 illustrates another characteristic of human background knowledge. The range of misclassifications—the “human error” alluded to in Chapter 1— indicates differences in individual’s background knowledge. Conversely, frequently misclassified articles, such as the twenty *soc.culture.czecho-slovak* articles mislabelled as *nz.general*, indicate common threads in the background knowledge of the participants as a group. The distribution of misclassifications might provide an interesting demographic study.

In general, most misclassifications were attributable to a small set of articles. Of the grid locations in Table 5.3 showing a frequency of at least 5, the majority refer to a single article; the exceptions being the twenty *czech* misclassifications—four articles, and the *nz.general* articles (group B) misclassified as *rec.arts.comics.strips* (group D)—three articles.

All of the articles misclassified by at least five people were *correctly* classified by TFIDF. The differences in weight between the newsgroups ranked first and second by the

algorithm were substantial. Inspection of these articles often reveals only one word in each that gives a definite hint as to the correct classification. Although these words are obviously identified by TFIDF, they are either missed or ignored by human classifiers. Interestingly, the C4.5 concept descriptions that misclassified these articles did so with the same classification as the majority of the human classifiers. This lends more weight to TFIDF's use of global word distribution information—distinctive words are often infrequent in their individual documents, but they exist nowhere else in the corpus.

Of the sixteen articles labelled incorrectly by TFIDF, ten were misclassified by at least one person. This shows the ambiguity present in many of the articles when they are observed out of context. Of the ten articles, C4.5 labelled eight incorrectly. These eight articles were the most difficult to classify—they were all misclassified by at least six people also. The other six consisted of five *rec.crafts.comics.strips* articles (group D), and one *nz.general* article (group B). Poor performance on the comics newsgroup was probably due to the small number (23) of training articles, compared to nearly 200 for the other groups. The reason for the lack of articles was the inclusion of the newsgroups FAQ. This article pushed the training data's word count up to that of the other groups, but when individual articles were separated again most of the words fell in this one article. If not for this one article amongst 1600 performance may have improved across the board.

The *nz.general* article was difficult for any algorithm or person unfamiliar with New Zealand and nature of the newsgroup. The article referred to the theft of a Ford Falcon car.

Stemming

Stemming had a small effect on classification accuracies with the general trend of a slight increase. Most of the schemes appeared to benefit from the reduced number of words. ZEROWORD and FIRSTWORD showed decreases of 7% in accuracy, however, although the compression rates of the two programs improved by 0.17 and 0.16 bits per character respectively. The compression abilities of these word-based schemes obviously benefit from having the myriad of suffixes removed from the text. Stemming reduced the number of unique words in the text from 27734 to 20663, reducing the size of the alphabet used for compression accordingly. The reduction in classification accuracy may support an hypothesis put forward by Lang (1995) that stemming removes information that word-based methods may be able to make inferences from. The other two word-based schemes, TFIDF and WORDAP, were not affected in the same way. These schemes gained accuracy with the stemmed text, although the increase was not as marked as the decrease for the other two algorithms.

The other three compression-based schemes all scored slightly higher in accuracy on the stemmed data. Compression rates for these programs remained unchanged on the stemmed text indicating that the removal of suffixes has little effect on alphabet size.

Data and concept description formats

No significant difference in accuracy is apparent between the ML schemes trained on frequency and appearance data. However the use of frequency data resulted in a significant decrease in the average number of rules generated by both schemes. Using numeric data the algorithms are able to specify high frequency thresholds for words that are very important to each newsgroup. The binary appearance data necessitates the use of more rules because the algorithms are unable to specify the relative importance of words to each newsgroup.

Making the task a ten-class problem instead of ten two-class problems results in a large increase in accuracy. C4.5 gains more discriminatory power from the combined set of words extracted from the ten newsgroups than from the ten individual sets. Although the single decision tree generated by this procedure is large, it produces a single classification for any test article. The combination of ten separate decision trees, one for each group, suffers from multiple-classification like the DNF rules of INDUCT.

Term selection methods

Results for the two methods of term selection are similar for the ten-class schemes indicating the selection method is not as important as simply providing a feature set that is representative of all classes. In the individual decision tree schemes, however, the difference in accuracy is substantial. The most-frequent-word term selection method places C4.5 in the middle of the accuracy rankings, whereas the 1R selected terms produce results little better than random guessing. There is no obvious explanation for this observation given the performance of the method in the ten-class task.

This aberration withstanding, there are no grounds for rejecting the most-frequent-word term selection method in favour of 1R. The former method is simple and fast, whereas 1R requires considerable preprocessing of the training data.

Compression-based schemes

The compression rates of the five compression algorithms generally agree with the findings of Moffat (1987)—the rate achieved by GZIP being better than expected. However, there does not appear to be any relationship between compression and classification abilities. This aspect is even more prominent in the stemmed data where word-based schemes gain a significant increase in compression rate but suffer a decrease in classification accuracy.

PPMC is easily the best compression-based algorithm for classification being at least 12% better than the next compression-based scheme in both experiments. This indicates that while it only placed third in terms of compression rate it is able to make better use of the similarities in the test article and its rightful newsgroup than the other algorithms.

6.

Conclusions

Machine learning is becoming an important source of tools for automating many information retrieval tasks. All phases of IR can be (and are) performed manually, but automation has many benefits—larger document collections can be processed more quickly and consistently, and new techniques can be easily implemented and tested. Human ability at these tasks provides an optimal performance threshold that many intelligent systems are approaching. However, perfect performance is probably unattainable while systems use such restricted models of natural language.

The most important phase of the information retrieval process is *feature selection*. This is particularly significant when machine learning algorithms are used at some point in the process. Although ML schemes perform feature selection themselves, they should initially be provided with a small number of distinctive features to learn from. Limiting the feature set in this way reduces space and time consumed by the algorithm, as well as lessening the risk of poor features being used in concept descriptions by chance.

Simple techniques for eliminating potentially weak features often work well once the nature of the features to be used is determined. Most IR systems use the word as the basic unit of content, and stoplists, stemmers, and thesauri help to substantially decrease the number of possible words a system has to cope with. When objects larger than words, such as phrases and paragraphs, are used as features, the need for studious selection is also important. Determining the extent of features such as these is possible using syntactic analysis, and sometimes through the use of formatting information. Coarser-grained features do not suffer from problems such as synonymy, so the co-occurrence of a phrase in two documents is a better indication of similarity than the appearance of the individual words of the phrase.

The *term-frequency–inverse-document-frequency* algorithm of document weighting is one of the simplest and most accurate methods for classifying text. It uses the term vector representation for documents suggesting that information about the distribution of important words in documents is often all that is necessary. The algorithm is resistant to the effects of stopwords and document length. Standard ML schemes using the same form of text representation are less resilient, requiring the data to be prefiltered to reduce the number of features. To reduce space and time consumption to practical levels the feature space must be limited to several hundred words. This is often a minute fraction of the total vocabulary of the text collection, so a good feature selection mechanism is necessary. The use of a *tf-idf*-style word selector may be profitable given the effectiveness of the algorithm for classification, although a simple most-frequent-words method may be sufficient. When the number of classes and the number of features are suitably limited machine learning algorithms generate concise and intuitive concept descriptions.

The use of ML techniques as the basis for specialist information retrieval applications has met with more success. Usually the "learning" is limited to a single phase of the IR process and the researchers have been able to concentrate on problems relating to that one area.

Four main phases are identified by Lewis (1991) and learning techniques have been applied to all of them. The first phase, *indexing*, incorporates feature extraction, document clustering, and text classification. Feature selection has already been noted as the most vital part of the process, and most researchers expend some effort in this area. Document clustering aids the IR system by creating links between similar documents. Clusters of related documents can then be retrieved once one of the documents has been deemed relevant to a query. Links may be made at any level, from common keywords to similar paragraph subtopics. By following links to an arbitrary level the system can retrieve documents to a given degree of similarity, and rank them accordingly.

The remaining three phases, *querying*, *comparison*, and *feedback*, usually form a loop, and are repeated until the user is happy with the retrieved documents. By making this an iterative process the system reduces the effort required of the user. Most users find it difficult to formulate their information needs in terms of keywords, and selection of "correct" words is a strenuous task. Systems that use feedback about the relevance of retrieved documents help the user by strengthening the weight of discriminatory terms present in the query, and introducing new terms selected from relevant documents.

The ease with which an effective query can be entered is enhanced by modeling the user. Creating a representation of the user's information needs reduces the effort required to select relevant documents in a changing information environment, and continual updating of the model, by means of user feedback, lets the system track shifts in the user's interests and information feed. By creating a model of each user's background and experience the system can interpret individual user's queries appropriately.

Machine learning technology has been successfully applied in many fielded and experimental information retrieval systems. The relative simplicity of the ML methods employed by system developers suggest that many of the problems faced by intelligent systems can be overcome using more advanced technology. Machine learning is a rapidly growing field, however, and new algorithms and techniques are continually pushing higher the limits of performance. Technology springing from such a rapidly advancing discipline is often obsolete by the time it is applied in real-world domains.

References

Allan, J. (1995) “Relevance Feedback with too much Data”, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seattle, Washington.

Allan, J. and Salton, G. (1993) “The Identification of Text Relations Using Automatic Hypertext Linking”, Department of Computer Science, Cornell University, Ithaca, New York.

Apte, C., Dameru, F. and Weiss, S.M. (1994a) “Automated Learning of Decision Rules for Text Categorisation”, *ACM Transactions on Information Systems* 12(3), pp. 233–250; July.

Apte, C., Dameru, F. and Weiss, S.M. (1994b) “Towards Language Independent Automated Learning of Text Categorisation Models”, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Dublin.

Baeza-Yates, R.S. (1992) “Introduction to Data Structures and Algorithms Related to Information Retrieval”, in *Information Retrieval: Datastructures and Algorithms*, William B. Frakes and Ricardo Baeza-Yates eds., Prentice Hall, Englewood Cliffs, New Jersey.

Bell, T.C., Cleary, J.G. and Witten, I.H. (1990) *Text Compression*. Prentice Hall, Englewood Cliffs, New Jersey.

Bhatia, S.J., Deogun, J.S. and Raghavan, V.V. (1995) “Conceptual Query Formulation and Retrieval”, *Journal of Intelligent Information Systems* 5(3), pp 183–209; November.

Bhuyan, J.N. and Raghavan, V.V. (1991) “A Probabilistic Retrieval Scheme for Cluster-based Adaptive Information Retrieval”, *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Illinois, pp. 240–244.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. (1984) *Classification and Regression Trees*. Wadsworth Inc., Belmont, California.

Cendrowska, J. (1987) “Prism: An Algorithm for Inducing Modular Rules”, *International Journal of Man–Machine Studies* 27(4), pp. 349–370.

Chen, H. (1995) “Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning, and Genetic Algorithms”, *Journal of the American Society for Information Science* 46(3), pp. 194–216.

Cleary, J.G. and Trigg, L.E. (1995) “K*: An Instance-based Learner Using an Entropic Distance Measure”, *Proceedings of the 12th International Conference on Machine Learning*, Tahoe City, California.

Cohen, W.W. (1995) “Text Categorisation and Relational Learning”, *Proceedings of the 12th International Conference on Machine Learning*, Tahoe City, California.

Conrad, J.G. and Utt, M.H. (1994) “A System for Discovering Relationships by Feature Extraction from Text Databases”, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Dublin..

Crawford, S.L., Fung, R.M., Appelbaum, L.A. and Tong, R.M. (1991) “Classification Trees for Information Retrieval”, *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Illinois, pp. 245–249.

Cunningham, S. and Summers, B.(1995) “Applying machine learning to subject classification and subject description for information retrieval”, Working Paper 95/20, Department of Computer Science, University of Waikato, Hamilton, New Zealand.

Frakes, W.B. (1992) “Introduction to Information Storage and Retrieval Systems”, in *Information Retrieval: Datastructures and Algorithms*, William B. Frakes and Ricardo Baeza-Yates eds., Prentice Hall, Englewood Cliffs, New Jersey.

Gaines, B.R. (1991) “The Tradeoff Between Knowledge and Data in Knowledge Acquisition”, in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W.J. Frawley, eds., AAAI Press, Menlo, California.

Garner, S.R., Cunningham, S., Holmes, G., Nevill-Manning, C., Witten, I.H. (1995) “Machine Learning in Practice: Experience with Agricultural Databases”, Working Paper 95/13, Department of Computer Science, University of Waikato, Hamilton, New Zealand.

Goker, A. and McCluskey, T.L. (1991) “Incremental Learning in a Probabilistic Information Retrieval System”, *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Illinois, pp. 255–259.

Haines, D. and Croft, W.B. (1993) “Relevance Feedback and Inference Networks”, *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, Pennsylvania.

Harman, D. (1992) “Relevance Feedback and Other Query Modification Techniques”, in *Information Retrieval: Datastructures and Algorithms*, William B. Frakes and Ricardo Baeza-Yates eds., Prentice Hall, Englewood Cliffs, New Jersey.

Hearst, M.A. (1993) “Cases as Structured Indexes for Full-Length Documents”, Computer Science Division (EECS), University of California, Berkeley, California, and Xerox Palo Alto Research Center.

Hearst, M.A. (1994) “Context and Structure in Automated Full-Text Information Access”, Technical Report No. UCB/CSD-94/836, Computer Science Division (EECS), University of California, Berkeley, California.

Holte, R. (1993) “Very simple classification rules perform well on most commonly used datasets”, *Machine Learning* 11 (1), pp. 63–90.

Kelly, G. (1963) *A Theory of Personality: The Psychology of Personal Constructs*. W.W. Norton and Co., New York, New York.

Krulwich, B. (1995) “Learning document category descriptions through the extraction of semantically significant phrases”, IJCAI Workshop on Data Engineering for Inductive Learning, at IJCAI-95, Montreal.

Lang, K. (1994) “NewsWeeder: An Adaptive Multi-User Text Filter”, Internal Research Report, Carnegie Mellon University; August.

Lang, K. (1995) “NewsWeeder: Learning to Filter Netnews”, *Proceedings of the 12th International Conference on Machine Learning*, Tahoe City, California.

Langley, P. and Simon, H.A. (1995) “Applications of Machine Learning and Rule Induction”, To appear in *Communications of the ACM*.

Lehnert, W., Soderland, S., Aronow, D., Feng, F. and Shmueli, A. (1995) “Inductive text classification for medical applications”, *Journal of Experimental and Theoretical Artificial Intelligence* 7, pp. 49–80.

Lewis, D.D. (1991) “Learning in Intelligent Information Retrieval”, *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Illinois, pp. 235–239.

Lewis, D.D. and Gale, W.A. (1994) “A Sequential Algorithm for Training Text Classifiers”, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Dublin.

Lewis, D.D. and Ringuette, M. (1994) “A Comparison of Two Learning Algorithms for Text Categorisation”, 3rd Annual Symposium on Document Analysis and Information Retrieval.

Lewis, D.D. and Croft, W.B. (1990) “Term Clustering of Syntactic Phrases”, *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, Brussels.

Martin, J.D. (1995) “Clustering Full Text Documents”, IJCAI Workshop on Data Engineering for Inductive Learning, at IJCAI-95, Montreal.

Moffat, A. (1987) “Word Based Text Compression”, Department of Computer Science, University of Melbourne, Parkville.

Quinlan, J.R. (1986) “Induction of Decision Trees”, *Machine Learning* 1, pp. 81–106.

Quinlan, J.R. (1990) “Learning Logical Definitions from Relations”, *Machine Learning* 5, pp. 239–266.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufman.

Ram, A. and Hunter, L. (1991) “A Goal-Based Approach to Intelligent Information Retrieval”, *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Illinois, pp. 265–269.

Rissanen, J. (1978) “Modeling by Shortest Data Description”, *Automatica* 14, pp. 465–471.

Robertson, S.E. and Sparck Jones, K. (1994) “Simple, proven approaches to text retrieval”, Technical Note, Department of Information Science, City University/Computer Laboratory, University of Cambridge; November.

Salton, G., Allan, J. and Buckley, C. (1994) “Automatic Structuring and Retrieval of Large Text Files”, *Communications of the ACM*; February.

Salzberg, S.L. (1990) *Learning with Nested Generalized Exemplars*. Kluwer Academic Publishers, Norwell, Massachusetts.

Seshardi, V., Weiss, S.M. and Sasisekharan, R. (1995) “Feature Extraction for Massive Data Mining”, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Montreal, pp. 258–262.

Soderland, S., Aronow, D., Fisher, D., Aseltine, J., Lehnert, W. (1995) “Machine Learning of Text Analysis Rules for Clinical Records”, Technical Report, Center for Intelligent Information Retrieval, Computer Science Department, University of Massachusetts, Amherst.

Soderland, S. and Lehnert, W. (1994) “Wrap-Up: a Trainable Discourse Module for Information Extraction”, *Journal of Artificial Intelligence Research* 2, pp. 131–158.

Soderland, S. and Lehnert, W. (1995) “Learning Domain-Specific Discourse Rules for Information Extraction”, AAAI 1995 Spring Symposium on Empirical Methods in Discourse Interpretation and Generation.

Towell, G., Voorhees, E.M., Gupta, N.K. and Johnson-Laird, B. (1995) “Learning Collection Fusion Strategies for Information Retrieval”, *Proceedings of the 12th International Conference on Machine Learning*, Tahoe City, California.

Utgoff, P.E. (1989) “Incremental induction of decision trees”, *Machine Learning* 4, pp. 161–186.

Witten, I.H. and Bell, T.C. (1990) “Source Models for Natural Language Text”, *International Journal of Man–Machine Studies* 32(5), pp. 545–579; May.

Witten, I.H., Moffat, A. and Bell, T.C. (1994) *Managing Gigabytes*. Van Nostrand Reinhold, New York, New York.

Appendix

Newsgroups used in text categorisation experiment

〈 *A* *alt.guitar*

Discussion of guitars, equipment and guitarists.

252 articles, 34815 words.

〈 *B* *nz.general*

Discussion of New Zealand related topics and current affairs.

177 articles, 35612 words.

〈 *C* *rec.arts.books.tolkien*

Discussion of J.R.R. Tolkien's books, characters and Middle Earth.

90 articles, 18956 words (this was all the articles in the group at the time).

〈 *D* *rec.arts.comics.strips*

Discussion of popular comic strips and artists.

22 articles, 35767 words (included the FAQ)

〈 *E* *rec.arts.startrek.current*

Discussion of current shows, characters and books on Star Trek.

196 articles, 37878 words.

〈 *F* *rec.crafts.brewing*

Discussion of techniques, equipment and materials for brewing beer, etc.

198 articles, 37988 words.

〈 *G* *rec.sport.cricket*

Discussion of current events in world cricket; scorecards.

186 articles, 35919 words.

〈 *H* *rec.sport.soccer*

Discussion of current events, teams and players in world soccer.

185 articles, 38004 words.

〈 *I* *sci.bio.paleontology*

Discussion of paleontology, scientific and informal.

181 articles, 36432 words.

〈 *J* *soc.culture.czecho-slovak*

Discussion of events and culture of former Czechoslovakia (in Czech, Slovak and English).

153 articles, 37883 words.

Figure A.1 Full text of test article *sci.bio.paleontology.07*.

Figure A.2 Lower case-only version of test article *sci.bio.paleontology.07*.

Figure A.3 Stemmed version of test article *sci.bio.paleontology.07*.

Example news article

Figure A.1 shows test article *sci.bio.paleontology.07* as it appears after saving from its newsgroup. Figure A.2 shows the same article after the header, numbers, and punctuation are removed, and all letters are converted to lower-case. All the structure has been removed from the article and it is now just a list of words. This is how the articles used in the first stage of the experiment appeared. Figure A.3 shows how the article appears after going through the PORTER stemming algorithm. This is how articles used in the second stage of the experiment appeared.

Description of schemes used in text categorisation experiment

COMPRESS

compress: UNIX Lempel–Ziv-based compression utility. Newsgroups are ranked on their ability to compress the test article; better compression means more closely related.

GZIP

gzip: Lempel–Ziv-based compression utility. Newsgroups are ranked on their ability to compress the test article; better compression means more closely related.

PPMC

ppmc.3: Character-based, context-modeling compression algorithm. Newsgroups are ranked on their ability to compress the test article; better compression means more closely related.

ZEROWORD

ZeroWord: Zero-order, word-based, context-modeling compression algorithm. Newsgroups are ranked on their ability to compress the test article; better compression = more closely related.

FIRSTWORD

FirstWord: First-order, word-based, context-modeling compression algorithm. Newsgroups are ranked on their ability to compress the test article; better compression = more closely related.

WORDAP

wordap: Poisson process model of word appearance. Determines expected number of novel words that will appear in test article. Groups are ranked on distance (in standard deviations) of actual number of new words from expected number. The implementation ignored non-letters and most punctuation, but did differentiate between upper and lower-case letters (these features were not used due to the way the data was pre-filtered). The program could perhaps use predictions of words appearing once, twice, etc. to further aid in prediction.

TFIDF

tfidf: Term frequency–inverse document frequency using cosine similarity measure. Term frequency vectors of newsgroup and test article are compared. Groups are ranked on difference between angles of normalised vectors. The implementation ignored non-letters and most punctuation, but did differentiate between upper and lower-case letters (again these features were not used due to the way the data was pre-filtered).

C4.5

c4.5: Creates a decision tree to classify examples. Rules are then synthesised from the decision tree, and used in “trickle down” manner, ie. once a classification is found no more rules are used. No multiple classifications can occur. Classification was treated as ten two-class problems, or one ten-class problem. In the two-class task to correctly an article the ten rule sets must all give the correct classification, ie. one positive classification and nine negative. Training data was pre-filtered using most-frequent-words or 1R to select a small number of attributes. The cutoff point was set to select at least sixty words, and this point sometimes covered several hundred.

INDUCT

induct: Creates a DNF rule set to classify examples. All rules are used during classification allowing for the possibility of multiple classifications for articles from each rule set. (This is the major reason that Induct has an accuracy of 64%. If a multiple classification contained the correct class the article was deemed to be classified correctly by PREVAL.) Classification was treated as two-class problem. To correctly classify an article the ten rule sets must all give the correct classification, ie. one positive classification and nine negative. Training data was pre-filtered using most-frequent-words to select a small number of attributes. The cutoff point was set to select at least sixty words, and this point sometimes covered several hundred.

HUMAN

homo sapiens: Carbon-based lifeforms consisting mostly of water. Utilises background knowledge to classify test articles. Training time in most cases of 20+ years. Probably the slowest method used for classification, and the most easily upset when informed of its performance.