Representation and Learning In Information Retrieval

David D. Lewis
Ph.D. Thesis
Department of Computer and Information Science
University of Massachusetts

COINS Technical Report 91-93
December 1991

REPRESENTATION AND LEARNING IN INFORMATION RETRIEVAL

A Dissertation Presented

by

DAVID DOLAN LEWIS

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 1992

Department of Computer and Information Science

Representation and Learning in Information Retrieval

A Dissertation Presented

by

David Dolan Lewis

Approved as to style and content by:

W. Bruce Croft, Chair of Committee

Gerard Salton, Member

Paul E. Utgoff, Member

F. Roger Higgins, Member

W. Richards Adrion, Department Chair
Computer and Information Science

ACKNOWLEDGMENTS

My greatest thanks go to Bruce Croft. His keen insight, high standards, and willingness to let me explore have had a huge impact both on this research and on my development as a researcher. I also thank my other dissertation committee members, Gerard Salton, Paul Utgoff, and Roger Higgins, who made substantial contributions from the perspectives of their respective fields.

Dozens of faculty, researchers, students, and staff in the Department of Computer Science at the University of Massachusetts, the Center for Information and Language Studies (CILS) at the University of Chicago, and elsewhere supported, prodded, enlightened, and aided me during the writing of this dissertation. My thanks to you all, and particularly to Howard Turtle and Robert Krovetz, fellow IR Lab students who have influenced this work in many ways.

John Brolio and Howard Turtle made their software available to me for these experiments and aided me in its use, for which I am quite grateful. Steve Harding and Peter Shoemaker wrote indexing software specifically for this research, and deserve special thanks. Raj Das also aided in these experiments. Panos Chrysanthis and Howard Turtle helped me sort out the mysteries of Latex. The Research Computing Facility at U Mass and the CILS systems staff provided help in numerous ways.

This research was supported by the Office of Naval Research under University Research Initiative Grant N0014-86-K-0764; by AFOSR under grant AFOSR-90-0110; by the National Science Foundation under grants IRI-8814790, IST-8414486, and an NSF Graduate Fellowship; by Sigma Xi, the Scientific Research Society; by the Defense Advanced Research Projects Agency under the Tipster initiative; and by the Ricoh and Olivetti corporations. DARPA, NSF, the American Association for Artificial Intelligence, the ACM Special Interest Group on Information Retrieval, and the Seventh International Conference on Machine Learning provided travel grants during the period of this research. The Longman Group made available the online version of the Longman Dictionary of Contemporary English. Phil Hayes, Carnegie Group, and Reuters made available the Reuters text categorization test collection. Ken Church and AT&T Bell Laboratories made available the *parts* program. Robert Morrissey, the director of CILS, gave remarkable support to an employee who devoted a disproportionate amount of time to his dissertation.

ABSTRACT

REPRESENTATION AND LEARNING IN INFORMATION RETRIEVAL

FEBRUARY 1992

DAVID DOLAN LEWIS

B.A., B.S., MICHIGAN STATE UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS

PH.D., UNIVERSITY OF MASSACHUSETTS

Directed by: Professor W. Bruce Croft

This dissertation introduces a new theoretical model for text classification systems, including systems for document retrieval, automated indexing, electronic mail filtering, and similar tasks. The Concept Learning model emphasizes the role of manual and automated feature selection and classifier formation in text classification. It enables drawing on results from statistics and machine learning in explaining the effectiveness of alternate representations of text, and specifies desirable characteristics of text representations.

The use of syntactic parsing to produce indexing phrases has been widely investigated as a possible route to better text representations. Experiments with syntactic phrase indexing, however, have never yielded significant improvements in text retrieval performance. The Concept Learning model suggests that the poor statistical characteristics of a syntactic indexing phrase representation negate its desirable semantic characteristics. The application of term clustering to this representation to improve its statistical properties while retaining its desirable meaning properties is proposed.

Standard term clustering strategies from information retrieval (IR), based on cooccurrence of indexing terms in documents or groups of documents, were tested on a syntactic indexing phrase representation. In experiments using a standard text retrieval test collection, small effectiveness improvements were obtained.

As a means of evaluating representation quality, a text retrieval test collection introduces a number of confounding factors. In contrast, the text categorization task allows much cleaner determination of text representation properties. In preparation for the use of text categorization to study text representation, a more effective and theoretically well-founded probabilistic text categorization algorithm was developed, building on work by Maron, Fuhr, and others.

Text categorization experiments supported a number of predictions of the Concept Learning model about properties of phrasal representations, including dimensionality properties not previously measured for text representations. However, in carefully controlled experiments using syntactic phrases produced by Church's stochastic bracketer, in conjunction with reciprocal nearest neighbor clustering, term clustering was found to produce essentially no improvement in the properties of the phrasal representation. New cluster analysis approaches are proposed to remedy the problems found in traditional term clustering methods.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this chapter, we discuss the range of tasks associated with computer-based access to textual information. We focus on text classification tasks, and in particular on the tasks of text retrieval and text categorization. Text classification tasks involve assigning pieces of text to content-based classes. The way in which texts are represented is a crucial influence on the effectiveness of systems for these tasks, but attempts to produce text representations enabling greater effectiveness mostly have been unsuccessful.

The lack of success of attempts to produce more effective text representations arises in part because current theories of text classification systems do not model the impact of text representations on the effectiveness of text classification systems. This means that most research on text representation has been driven by informal intuitions about the nature of human language.

The goal of this dissertation is to provide a more solid foundation for research on text representation for text classification. In particular, we propose:

1. To outline a theoretical model that enables the behavior of text classification systems to be in part predicted from measurable characteristics of text representations.

2. To test the predictions of this model on a text representation whose behavior has been difficult to understand under existing models of text classification.

The formulation of models that make testable predictions about representation quality should eventually enable the construction of text representations supporting more effective text classification systems.

Besides introducing the issue of text representation for text classification systems, this chapter outlines our investigation of this issue, and how the results of that investigation are presented in this dissertation.

1.1 Content-Based Text Processing

By almost any measure, the amount of information being produced is growing faster than the ability of information consumers to find, digest, and use this information. One response has been to publish information in computer-accessible form rather than by traditional media such as paper, film, audio tape, and video tape. Businesses and other organizations store an increasing amount of their internally generated information in computer-accessible form. A small but increasing

proportion of both technical and everyday correspondence and conversation occurs, and is recorded, by electronic mail and voice mail, adding to the opportunities and problems created by information growth.

Merely recording information on computer-readable media is only the beginning of a solution to information overload. Users need methods of finding the particular information they want, and examining it in its original or a summarized form. For some kinds of information and some kinds of information needs, this is relatively straightforward. For instance, database software, spreadsheets, statistics packages, and graphical display methods have made access to many kinds of numeric data quite easy.

It is much harder to provide effective access to data that consists of expressions in human language. This includes technical articles, memos, manuals, electronic mail, books, newspapers, magazines, journals, and many other forms of text. In addition, it is desirable to access other forms of data, including speech, images, video, financial data, and chemical structures, through textual annotations to these complex objects. Textual data is difficult to access, however, because the relationship between its form (typically sequences of characters) and its content is less clear than, for instance, in numeric data.

Content-based text processing tasks can be divided into two broad groups. *Text classification* involves the assigning of documents or parts of documents to one or more of a number of groups. *Text understanding* involves more complex access to the content of documents, such as extracting formatted data, answering questions, and summarization or abstracting.

The division between these tasks is not sharp. In particular, software components that accomplish text classification are often embedded in systems for more complex text processing tasks. This makes text classification, which has been the focus of *information retrieval* (IR) researchers and which will be the focus of this dissertation, of particular interest.

1.2 Text Classification Tasks

"Classification" is an ambiguous term in information retrieval, applied statistics, psychology, and other fields but almost always refers to processes of grouping of entities. Text classification therefore is an appropriate term to subsume a number of information retrieval tasks that are usually considered distinct, but which all involve grouping of textual entities. We describe seven such tasks in this section, focusing on the two that have received the most attention in information retrieval: text retrieval and text categorization. After this, we examine the similarities and differences between the tasks and the role that the representation of text plays in them.

1.2.1 Text Retrieval

Text retrieval is the computer selection of a subset of a document database to display in whole or summary form to a user, usually in response to a user request.

One view of a text retrieval system is that it sorts documents into two classes: documents that will be displayed to the user, and those that will not. Many advanced text retrieval systems not only select documents for display, but also attempt to order displayed documents by importance. These systems can be viewed as computing the degree of membership of documents in a class without sharp boundaries.

We can describe the text retrieval process as consisting of four main phases [1]:

1. *Indexing:* Raw documents must be converted into expressions in some text representation. These expressions are sometimes called *document representatives*, and must have a structure usable by the text retrieval software.

2. *Query formulation:* The user must express his or her information need in the form of a *request* interpretable by the IR software. The request is sometimes entered by the user in a form very similar to that used by the system, such as a boolean expression over words. In other cases the connection may be less direct, as when the user enters a natural language question or example document and the IR software selects important words from the user input to use as features in a statistical classifier. We use the term *query* to refer to the form of the user request actually compared to documents, though this will not always be an easily identifiable data structure in a text retrieval system.

3. *Comparison:* The system must implicitly or explicitly compare the user query to the stored documents, and make a classification decision about which documents to retrieve and in what order. Documents or parts of documents are displayed to the user.

4. *Feedback:* An initial retrieval rarely results in exactly the documents desired by a user. Several iterations of modifying the query are often necessary to achieve acceptable results. This modification may be done explicitly by the user, if he or she chooses to enter a new request or modify their original one. On the other hand, the user may simply communicate to the text retrieval system judgments about the desirability of each retrieved document, and the system may implicitly update the query. This latter process is referred to as *relevance feedback.*

1.2.2 Text Categorization

Text categorization is the classification of documents with respect to a set of one or more pre-existing categories. The most common application of text categorization is in indexing documents for text retrieval, i.e. in producing document representatives. Manual assignment of subject categories to documents is a widely used form of text representation. Users can mention these subject categories in their requests, possibly enabling a more compact and effective query to be formed. However, manual assignment of categories requires considerable human labor and

[1] This model is due to Bruce Croft.

expense. Replacing or aiding manual indexing with automated text categorization can reduce these costs substantially. Several such systems are operational, in areas such as indexing of newswire stories for subject specific distribution [HW90], and indexing of large databases of technical abstracts [BFL+88, FHL+91].

Another application of text categorization is within text understanding systems. Categorization may be used to filter out documents or parts of documents that are unlikely to contain extractable data, without incurring the costs of more complex natural language processing [DLW+91, GSM91, Hob91]. Parts of a data extraction task can sometimes be performed by text categorization alone [BT91, DGCN91, DR91, Lew91a]. In a system that processes texts in many subject areas, categorization may be used to route stories to category specific processing mechanisms [DeJ82, JR90].

Finally, the categorization itself may be of direct interest to a human user, as in judging whether a threatening letter against a government official signifies real danger [Har88a].

As with text retrieval, a category may be binary (a document either is or is not a member of the category) or graded (a document can have a degree of membership in the category). Binary assignments have been used in most applications. When multiple categories are used, it may be the case that each document is assigned to exactly one category. On the other hand categories may be assigned independently, with each document falling into all, some, or no categories.

Text categorization can be described as consisting of the same phases as text retrieval, though some details are significantly different:

1. *Indexing:* The same text representation techniques used in text retrieval can be used here. Speed of indexing is often more critical than in text retrieval, because large numbers of new documents may need to be processed in real time.

2. *Categorizer Formulation:* Like text retrieval, text categorization requires a specification of how to decide to which categories a document should be assigned, based on its text representation structures. For text categorization systems we will call these internal specifications *categorizers*. They play the same role as queries in a text retrieval system.

   While text retrieval queries are typically temporary structures, resulting from an ad hoc user request, a set of categorizers will often be in use over a long period of time. Substantial expert effort or large scale statistical analysis may be used in their construction and testing.

3. *Comparison:* In most text categorization systems, a binary decision is required from each categorizer about each document. The most important difference from text retrieval systems is that a document may be compared to a number of categories at once, and the appropriate decisions may depend on relationships among categories.

4. *Adaptation:* Feedback plays two roles in text categorization systems, both somewhat different from its role in text retrieval. First, when a categorization system is constructed, there are often large numbers of manually categorized documents already available. Automated techniques similar to relevance feedback can use these examples to construct or aid the construction of categorizers. Second, while users of the categorized documents are typically not allowed to provide feedback to alter the categorization system, they can communicate their reactions to the categorization system maintainers, who may modify, add, or remove categories and associated categorizers.

1.2.3 Text Routing

Text routing, also known as *selective dissemination of information*, *SDI*, or *text filtering* combines aspects of text retrieval and text categorization. Like text categorization, a text routing system processes documents in real time and assigns them to zero or more of a set of classes. However, like text retrieval, each class is typically associated with the information needs of one or a small group of users [Col81]. Each user or user group can typically add, remove, or modify the standing queries or *profiles* associated with their needs. There may or may not be relationships among profiles, and profiles may or may not be under end user control. Relevance feedback can be used in text routing and has the potential for being more effective than in text retrieval, since the information need persists over a longer period of time.

1.2.4 Term Categorization

Term categorization is similar to text categorization, in that pieces of text are assigned to predefined categories. The difference is the size of the pieces of text. Where text categorization deals with full documents, or relatively large portions of documents, term categorization is the assignment of categories to words or small fragments of text. While there is some blurring of this task into text categorization, the techniques applied are different enough to consider this a distinct task. Applications of term categorization include *tagging* (the assignment of syntactic or semantic categories to words to support natural language analysis) and automated assignment of free-indexing phrases [Fie75, RJ91]. (We describe tagging in more detail in Section 10.3.2.)

1.2.5 Document Clustering

The final three tasks we will describe differ from the first four in involving not only the assignment of portions of text to categories, but the creation of those categories from a corpus of text. *Document clustering* is the automated generation of categories of documents, usually based on some similarity measure between documents, as well as a definition (explicit or implicit) of what characteristics groups of documents should have. Document clustering has been suggested both as a means to speed up physical access to stored documents and as a text representation for

improving the effectiveness of text retrieval. In both roles it has provided mixed results [Wil88].

1.2.6 Term Clustering

Term clustering is similar to document clustering, except that individual words or small fragments consisting of closely connected words are formed into groups. Term clustering is described in more detail in Chapter 4. Term clustering has been investigated for producing better text representations to support text retrieval, so far without much success. It has also been used as a method for studying word usage and producing information useful to natural language processing [Hin90].

1.2.7 Latent Indexing

Latent indexing is related to both term clustering and document clustering. It uses factor analysis or related techniques to transform one representation of a collection of documents into a new representation with desirable mathematical properties. Both the original indexing terms and the original documents are reexpressed in terms of this new representation. This reexpression is meant to support improved text retrieval [DDF+90], though the technique has not led to significant effectiveness improvements and is computationally expensive.

1.2.8 Summary

Of the tasks discussed, text retrieval, text categorization, and text routing are the most directly applicable to human information needs. In contrast, the last four tasks are mostly performed in support of other text processing tasks. The first three tasks differ considerably in how classifiers (queries, categorizers, or profiles) are formed, but all require that classifiers are formed and that documents are represented in such a way that an effective comparision can be made between the classifier and documents. Text representations play a related role in document clustering and latent indexing.

The emphasis in this dissertation will be on the text retrieval and text categorization tasks, but many of the results produced will be directly relevant to text routing, document clustering, and latent indexing, and indirectly relevant to term clustering and term categorization.

1.3 Text Representation

The comparison in the previous section showed that text representation plays a consistent and important role in a variety of text classification tasks, including the three tasks most widely performed in operational settings: text retrieval, text categorization, and text routing. In this section we examine current beliefs about text representation in information retrieval, and consider two paradoxes with respect to the issue of representation quality.

Investigations of different text representations in information retrieval has been extensive, particularly with respect to their effect on text retrieval effectiveness. Reviews of this research have led to the widely held conclusion that no text representation is significantly superior to representing documents by isolated words drawn from the original text, and that text representation has a relatively minor influence on effectiveness:

- "...artificial indexing languages do not perform strikingly better than natural language...complex structured descriptions do not perform strikingly better than simple ones...the characterization of queries is more important than that of documents..." [Spa81]
- "No support is found in the literature for the claim that text-based retrieval systems are inferior to conventional systems based on intellectual human input." [Sal86]
- "Index languages uncontrolled at the indexing stage do not have an inferior performance to controlled ones...Syntactical devices used explicitly in searching (e.g. links, roles, relations) as improvers of precision have a small and minority influence...the index language vocabulary has a minor influence on performance compared with query negotiation, searching, and indexing..." [Kee81]
- "Representation of documents with complex, controlled languages that incorporate phrases, thesaurus, and other structural relationships has never been shown to be effective in terms of retrieval performance." [Cro87]

These pessimistic results can be constrasted with the successes enjoyed by research on acquiring better requests from users [CD90], mathematical models for queries [BC87], and relevance feedback techniques for improving queries [CD90, SB90].

The failure of human effort to produce better text representations is surprising. It is easy to imagine terrible text representations that would support no better than random classification of documents. It is easy to imagine an excellent text representation that happens to contain a single term indicating exactly the set of documents specified by the current user. Yet all reasonable text representations have been found to result in very similar effectiveness on the text retrieval task. We call this the *Equal Effectiveness Paradox.*

Another paradox arises from the fact that the effectiveness of current text retrieval systems, while good enough to make them of practical use, is far from perfect. Yet most text representations and text retrieval systems in use have the property that, given almost any subset of the database, it is possible to create a request that will be translated into a query retrieving exactly those documents. This is true because, with most text representations in use today, any document can be uniquely identified by a few terms that occur in few or no other documents. We call this the *Perfect Query Paradox.*

In one sense there is nothing mysterious here. It is not surprising that users do not enter perfect requests, which might require taking advantage of knowledge of

the peculiarities of a particular document collection. However, the possibility that a user could do this raises the question of whether it is possible to discuss the quality of text representations independent of some psychological model of users.

The failure of empirical attempts to produce better text representations, and the Perfect Query Paradox, suggest that a better theoretical understanding of text representation is needed. Such a theory may not be sufficient to produce better text representations, but it is likely to be necessary. This dissertation is a small step towards such a theory.

1.4 Effectiveness in Text Classification

We have repeatedly referred to text representations providing better or worse text classification effectiveness, without saying what effectiveness means for text classification. For the tasks on which we are focusing (text retrieval and text categorization), the primary question we have about a system is how well it can assign documents to the correct classes. In text retrieval the class of interest is the class of documents relevant to the user's information need. In text categorization there may be several classes of interest, for instance a group of subject categories.

Consider a single class, and suppose that the text classification system judges each of a set of documents to belong to the class or not belong to the class. The system can err in either assigning documents to the class when they are not members, or in failing to assign documents to the class when they are members. Two important measures of system effectiveness are *recall* and *precision*. Recall is the fraction of all documents belonging to a class that were assigned by a system to that class. Precision is the fraction of documents assigned to the class which actually belonged to the class.

Often text classification systems have parameters that can be varied to increase or decrease their tendency to assign documents to a class. Such systems usually exhibit a tradeoff between recall and precision—recall can be increased only by decreasing precision. It is often desirable to display the recall and precision values for a text classification system at several parameter levels to see how this tradeoff varies.

We discuss effectiveness measures in more detail in Section 6.4, emphasizing the text categorization case since it has received less attention in the past.

1.5 Outline of the Dissertation

This dissertation is in three main parts, one theoretical and two experimental. The first part, Chapters 2 through 4, addresses the first of the goals set forth in the introduction:

1. To outline a theoretical model that enables the behavior of text classification systems to be in part predicted from measurable characteristics of text representations.

Chapter 2 is a survey of research on the role of representation in classification systems. We focus on systems where classifiers are automatically induced from training data, as in relevance feedback and in the automatic training of text categorization systems.

In Chapter 3 we turn our survey to previous work on text representation in information retrieval. We examine previous attempts to provide theoretical models for the quality of indexing terms and sets of indexing terms. These models have not, unfortunately, led to better text representations and most have not even made testable predictions about representation quality. We present the beginnings of a new model of text classification systems, the concept learning model (CL model). The core of this model is the claim that there are properties of text representations that impact the effectiveness of text classifiers whether the classifiers are built by automated or manual means.

Finally, Chapter 4 examines a particular text representation, syntactic phrase indexing, in the light of the CL model. Despite the considerable capabilities of current syntactic analysis technology, attempts to apply syntactic parsing to text retrieval systems have yielded consistently disappointing results. The CL model suggests why this has been the case, and we propose a possible way to improve the quality of this representation by following syntactic phrase indexing with term clustering.

The second and third parts of the dissertation address our second objective:

2. To test the predictions of this model on a text representation whose behavior has been difficult to understand under existing models of text classification.

The second part of the dissertation is Chapter 5, which presents a test of syntactic phrase clustering on a standard text retrieval test collection. The result was a minor improvement in effectiveness. We also uncovered a number confounding factors that make it difficult to interpret the impact of text representation on text retrieval.

The third part of the dissertation is a more extensive investigation of the quality of syntactic phrase clusters and related text representations. We begin in Chapter 6 by showing how text categorization has some desirable properties as an alternative task for studying text representation. We consider a number of potential drawbacks, but conclude that further investigation of syntactic phrase clusters would be best conducted using a text categorization task.

Chapter 7 describes the software system, Maxcat, that we constructed for our experiments on text categorization. We emphasize those aspects of Maxcat's design that enable increased flexibility in studying text representations.

While having many desirable properties from the standpoint of studying text representation, text categorization had the disadvantage of being less studied and less well understood than text retrieval. Before text categorization could be used to study text representation quality, it was necessary to have a text categorization procedure whose effectiveness we trusted. In Chapter 8 we replicate the text categorization method used in an early and widely cited study on categorization and

uncover a number of disadvantages of the method. Chapter 9 presents results on a new categorization method that remedies some of these difficulities and achieves considerably better, though not ideal, results.

In Chapter 10 we return to syntactic phrase clustering, and present the results of experiments using syntactic phrase clusters and closely related text representations for text categorization. A number of effects predicted for text representations by the CL model were found to occur. However, statistical term clustering was not found to improve the quality of a syntactic phrase representation in the text categorization task. Given the careful controls on our experiment, we conclude with some confidence that a range of statistical term clustering methods are unlikely to provide significant improvements in the quality of a syntactic indexing phrase representation. Some evidence is presented to suggest that alternate approaches to clustering would yield an improved text representation.

Finally, Chapter 11 summarizes the theoretical and experimental results of the dissertation and suggests directions for future research on text representation and text classification.

CHAPTER 2

REPRESENTATION IN CLASSIFICATION

In this chapter we briefly survey machine learning of classifiers, i.e. functions that can be used to assign objects to one or more of a set of preexisting classes. We focus on how the features used to represent objects affect learning of classifiers, and how feature sets can improved. Our goal in this chapter is to pick out the ideas from machine learning research that are most important to our examination of text representation.

2.1 Terminology

The task of assigning objects to one or more of a set of preexisting categories has been studied in many different disciplines and so is called by many different names. The most common terms in research that emerges from the statistics community seem to be *classification* or *classification analysis* [Jam85, BFOS84] and *discriminant analysis* [GD78, Lac75]. Unfortunately, both "discrimination" and "classification" have been widely used with meanings that contradict this one. Classification is at least as widely used with a meaning synonymous or nearly synonymous with *cluster analysis* [SJ67, Gor81], which does not even involve preexisting classes. In research by the artificial intelligence community the task is often called *concept learning* [MCM86], which itself has a distinct meaning in psychology.

Hand [Han81] uses the term *discrimination* to refer to the process of deriving classification rules, and the term *classification* for applying those rules to new objects. The distinction is relatively clear, even if the terminology is not. In describing the structure of learning systems, Weiss and Kulikowski distinguish between the *learning system*, and the *classifier* that the learning system builds; the classifier makes the decision of what class or classes to which an object should be assigned [WK91]. For the purposes of this study we will use the term *learning* or *supervised learning* to refer to the process of mechanically building a *classifier* for a set of known classes, and *classification* to the process of using the classifier. (Note that the term "text classification," as defined in the previous chapter, includes the application of techniques besides supervised learning.) The term *learning* sometimes subsumes techniques like cluster analysis, which discover new classes, but we will always use the term *clustering* or, more generally, *unsupervised learning* for the discovery of new classes. We will use the term *machine learning* to refer to the study of both supervised and unsupervised learning.

Distinguishing learning from classification is important, since classifiers can be built by human knowledge engineering, without the using of machine learning techniques ([Cla85]; [WK91], ch. 7).

We use the term classifier to refer both to a mathematical function and to instantiations, potentially imperfect, of that function in computer programs. The domain of a classifier function is the space of all possible inputs that might be given to the classifier. We will assume that each input consists of a ordered tuple of values, and that the value on each dimension of the tuple is derived in some fashion from the object to be classified. These dimensions go by many names, including *features*, *attributes*, *observations*, *variables*, *tests*, and *measurements*. We will typically use the term *feature*. The desired outputs of a classifier, i.e. the decisions about class membership, are sometimes referred as *criterion features*, and when using this term we will call the input features *predictor features*.

The range of a classifier is the set of possible specifications of classes to which an input can belong. We will assume that, if we have n classes, the range is the space of all n-tuples of such specifications. Typically such specifications are either binary (*True* vs. *False*, or 0 vs. 1), or real-valued weights indicating a probability or degree of membership in a class. The case where classifications can be real-valued blends into the related task of regression analysis, which is concerned with predicting real value quantities based on observations. We will call a particular input to a classifier an *instance* and a particular output a *classification vector*.

2.2 Forming Classifiers by Machine Learning

In mathematical statistics, it may be of interest to prove that there exists a classifier function with certain properties, without necessarily displaying it. For applications, however, we need explicit examples of such functions, typically instantiated in the form of computer programs. Classifiers can be built by a variety of means, including human design with no use of mechanical methods, as in expert systems. It is often desirable, however, to replace or augment human effort by mechanical means. A common division of effort is for the human engineer to create a set of features by which objects will be described, and then use machine learning to find a function that approximates the desired mapping from feature vectors to classification vectors.

There are a variety of methods by which a machine learning system can construct a classifier. Most of these methods involve making available to the program a set of *training instances*: a set of feature vectors paired with the desired classification vectors. The learning program then attempts to find a classification function that in some specified fashion agrees with the training instances of the desired mapping. The hope is that the function found will reproduce the desired mapping not just on the training instances, but on previously unseen *test instances*.

The process by which a learning program produces a classifier function varies greatly from method to method and program to program. In work on learning in the artificial intelligence community, this process is often treated as one of *search* [Mit82]. The program is viewed as considering candidate functions from an implicit

or explicit *hypothesis space*, evaluating them in some fashion, and choosing one that meets some criterion. If one is willing to allow sufficiently implicit representations of the hypothesis space and sufficiently loose definitions of evaluating alternatives, even learning methods that are based on estimating parameters of a probability distribution or fitting a curve to points can be viewed as search.

The hypothesis space that a learning program explores is typically determined by two main factors. One is the set of features used to describe objects. It is obviously pointless for the learning algorithm to explore functions that depend on features of instances that will not be available when the classifier is used. However, learning programs will sometimes construct new features defined in terms of the initial features (see Section 2.4.2).

The second factor is the class of functions that the program can consider, sometimes called the *form*, the *model*, or the *language bias* [Hau88]. The class of functions is often defined explicitly and in terms of mathematical properties of the functions, such as a restriction to linear polynomials or disjunctive normal form over boolean variables.

2.3 The Quality of Feature Sets

Typically there are many possible hypotheses that agree with a set of training data, so machine learning algorithms need additional constraints to choose one of these hypotheses. The set of all these constraints is the *bias* of the learning algorithm [Utg86]. We have already mentioned the language bias, the part of the bias introduced by the class of functions an algorithm is willing to consider. Given our focus on text representation, we are interested in a different bias—the one that results from the set of features used to describe objects—and how this bias can interfere with learning. In the following we discuss a number of ways in which a feature set may make learning a classifier difficult or impossible.

2.3.1 Feature Set Does Not Sufficiently Distinguish Instances

An initial set of features has been defined to be *epistemologically adequate* if it allows every possible instance to be distinguished [Ren88]. Quinlan gives a similar definition but uses just the term *adequate* [Qui86b].

Clearly if the feature set is not epistemologically adequate, then it may not be possible to learn the desired function with 100% accuracy. Some distinct instances will appear the same to the learning system, and the desired function may take on different values for those instances. Of course, in order to say that a set of features is not epistemologically adequate, we must ourselves be using some additional information to distinguish instances that are not distinguished by the available features, and this additional information potentially could be used to create new features.

2.3.2 Feature Set Excludes Concept from Hypothesis Space

A set of epistemologically adequate features means that some function exists that implements the desired concept. However, the combination of the feature

set and the language bias may result in a hypothesis space that does not contain the concept, or does not contain a function that approximates the concept to the desired accuracy. For instance, the learning system might be searching a space of linear classifiers, but there may be no hyperplane that separates the positive from negative instances in the space defined by the features.

Note that given a sufficiently good set of features, almost any language bias will result in a hypothesis space that contains the desired concept. In particular, if some single feature already implements the desired concept, any language bias that allows that feature to be used will allow a perfect classifier.

2.3.3 Feature Set Results in "Too Big" a Hypothesis Space

The effect of the number of features on the ability of a learning algorithm to find an accurate classifier has been widely investigated. One's first intuition is that more features carry more information and must always result in a more accurate classifier. A less obvious counterbalancing intuition is that more features increase the number of hypotheses and make it harder to find an accurate classifier. The problem of having too many features is sometimes refered to as "the curse of dimensionality" ([DH73], p. 95).

The true situation is more complicated than either of these intuitions, and is by no means completely understood. Jain and Chandrasekaran survey a large body of theoretical and empirical work in statistical pattern classification showing that, for a fixed number of training instances, there is often a maximum number of features beyond which effectiveness of the induced classifier starts to decline [JC82]. They cite as a general rule of practice that the number of training instances should be five to ten times the number of features. Chatfield suggests at least four times as many training instances as features when fitting multiple linear regression models, and prefers limiting the number of features to four or five ([Cha88], p. 199). Banerji [Ban71] cites an example where the sample size must be 20 times the dimensionality of the space to achieve 95% accuracy in training a classifier. For a text classification task, Fuhr suggests 50–100 samples per parameter are needed [FB90].

A very rich line of theoretical research has emerged from a formalization by Valiant [Val84] of the intuitive ideas of hypothesis space "size", "ability to find hypothesis", and "accurate classifier" we mentioned above. Results can be derived under the Valiant model for the minimum number of training instances necessary to learn concepts with various language biases, under suitable assumptions [Hau88]. All these results show the number of training instances needed to rise with the number of features. Results under the Valiant model are worst-case estimates and deviate substantially from numbers of samples required for real world applications. However, they provide some guidance, and attempts have been made toward producing more empirically useful estimates [Paz90, Kad91].

2.3.4 Feature Set Violates Assumptions of Search Algorithm

A feature set may result in the desired concept being present in the hypothesis space, but may make it difficult or impossible for a search algorithm to find the

desired function. A wide variety of assumptions are made by different algorithms and can be violated:

- A classic method by Fisher [Fis36] forms the optimal linear discriminant function (given known population parameters) only when the feature set is such that a concept and its inverse have a multivariate normal distribution, with equal covariance matrices, over the space defined by the feature set ([Jam85], p. 29). In practice, the method tends to find reasonable classifiers even under deviations from normality ([Jam85], p. 61). The corresponding optimal method for multivariate normal distributions with unequal covariance matrices is much less robust to deviations from normality ([Lac75], p. 20; [Jam85], p. 61). Other sorts of distributional assumptions can be made, and violated, as well, such as independence of conditional probabilities (see Chapter 8).

- Many learning algorithms, for example those used with connectionist networks, make use of gradient descent or hill-climbing through the space of hypotheses ([WK91], ch. 4). These methods are guided by some direct or indirect measure of the quality of hypotheses. If the feature set is such that the quality measure has local minima on the hypothesis space, then the search procedure may terminate before finding the desired function.

- Many empirical learning methods assume that the desired function is continuous or nearly continuous on the instance space and are harmed by feature sets that result in discontinuous concept functions [Mat91]. Similar sorts of assumptions, such as assuming a relatively small perimeter for the concept in the instance space [KA88] can also lead to problems when violated.

Many other examples could be given. From one standpoint, a characteristic of a feature set is a problem exactly because it violates some assumption, implicit or explicit, of commonly used learning algorithms. The examples listed above are just cases where the assumptions are more explicit and well-defined than usual.

2.3.5 Feature Set Is Noisy

Features may be noisy. In statistical communication theory [Sel65, Ham80], the concept of noise is well defined. The noise introduced by the physical properties of a communication channel or of a sensor device can often be modeled quite accurately as a random process added to the true signal.

For most machine learning applications, the concept of noise is much less well defined. Quinlan [Qui86b] talks extensively about the effect of noise on learning, but does not directly define the term, instead giving this description:

> In real-world classification tasks, the description of an object will often contain errors. Some sources of these errors are faulty measurement, ill-defined thresholds (e.g. when is a person "tall"?), and subjective interpretation of a multitude of inputs (e.g., what criteria are used when describing a person as "athletic"?).

Weiss and Kulikowsi's text ([WK91], p. 11) is similarly informal:

> Those features that are no more predictive than chance can be considered *noise*. For any given application, features will probably range from those that are completely noisy, to somewhat noisy, to completely predictive. Features that appear noisy on their own may prove to be highly predictive when combined with other features.

Whatever the definition, noise can lead a learning mechanism to induce a different, usually poorer classifier than it would from noise-free data, and can reduce the accuracy of any classifier when applied to new instances.

Closely related to noise is the problem of missing values. The value of a feature may not be available for all training or test instances. It may sometimes be known that data is missing, but missing data can also manifest itself as noise. A feature may return a default value such as 0, for instance, rather than explicitly indicating that no data was available. If the value 0 has some additional meaning besides indicating missing data, missing data appears as noise in feature values. Most learning algorithms assume that all feature values are available, requiring a variety of special fixes to be developed to handle missing values ([WK91], pp. 172–173).

2.3.6 Feature Set May Contain Redundancy

Features that appear to be distinct may actually be measurements of the same underlying property of an object. How serious this problem is depends on the learning algorithm. For instance, redundant features can make the distance measures used by nearest neighbor methods misleading ([WK91], p. 73). On the other hand, learning methods that incorporate feature selection (see below) may be able to detect and ignore redundant features in some cases [Qui86b]. Redundancy can even be desirable, since when treated appropriately it can help correct for missing values or noise ([WK91], p. 73; [Dra91]).

2.3.7 Other Problems

There are many other ways, mostly poorly understood, in which feature sets can deviate from ideal. Often these problems are described in terms of their effect on the shape of desired function over the instance space. We mentioned earlier that feature sets that cause the target function to be discontinuous over the feature space violate the search bias of many learning algorithms. Stating the problem in this way tends to make one think in terms of finding an algorithm with a different search bias. We can always imagine an algorithm that just happens to find the desired hypothesis, no matter how bizarre the form of the hypothesis, given the current feature space. Yet clearly this will not happen in real learning systems. One is left with a strong intuition that some feature sets are bad for all sensible learning algorithms. Connections between data compression, a simplicity bias in the order in which learning algorithms explore hypotheses, and computational complexity theory may be useful in formalizing this intuition [Sax91].

2.4 Improving Feature Sets

In the previous section we discussed ways in which the set of features used to represent instances can adversely affect the ability of a learning program to find a function that accurately classifies new instances. This sensitivity of learning systems to instance representation has led researchers to develop methods of improving representations. We can broadly classify these strategies into two main groups: *feature selection* and *feature extraction* [Kit86]. A great deal of research has been done in both areas, and we present only a relatively brief survey of it below.

2.4.1 Feature Selection

Feature selection involves choosing some subset of d features from an original set of D features, where $d < D$.[1] In almost any practical classification problem, considerable feature selection will have been done by human beings in choosing measurements to report that are likely to be of use in classification. However, it is equally true that the best subset of features is often not readily apparent, so that automated methods of feature selection are useful. In addition, if new features are created by the learning algorithm during learning, then automated selection among these may be needed.

Automated feature selection requires the ability to evaluate the quality of individual features or subsets of features. The intuitive quality measure is the error rate of the classifier on unseen instances, or perhaps the expected cost of errors. Estimating these measures, however, is computationally expensive and may require more test samples than are available ([Jam85], p. 127). Even when the probability of error can be estimated, it may not be an ideal quality measure for features or feature sets [Ben82].

A wide range of other measures, based on overlap of probability density functions, statistical dependence, information-theoretic measures, distance measures, cluster analysis, dispersion measures, and others have been suggested ([Kit86]; [Jam85], ch. 8; [JD78]). In examinations of algorithms for inducing decision trees, Breiman, et al ([BFOS84], p. 38) and Mingers [Min89] found that a wide range of feature selection algorithms yielded similar classification effectiveness. Ben-Bassat [Ben82] cites results where a variety of feature evaluation methods were found to produce similar feature rankings, and suggests that computational efficiency be a primary consideration.

Once a feature quality measure is selected, it is then necessary to evaluate potential subsets of the feature set. Since there are $D!/(D-d)!d!$ subsets of size d, exhaustive search is usually not practical, and heuristic techniques or assumptions about the independence of feature quality are necessary. The simplest approach is to evaluate the quality of each individual feature and pick the d best. This

[1] We take our usage of the terms feature selection and feature extraction from Kittler's survey [Kit86]. Other authors have defined these terms differently or used other terms to describe the same or related procedures.

does not ensure good effectiveness of the resulting subset of d features, even when features are known to be conditionally independent [Ben82]. Stepwise procedures, which incrementally add features, discard features, or both, evaluating the subset of features that would be produced by each proposed change, are more effective ([Jam85], pp. 127–129; [Kit86]; [WK91], pp. 74–75). Feature selection can be done before training, during training, and even during classification, particularly when incremental learning algorithms are used.

Evaluation of feature sets during feature selection not only increases the effectiveness of classification, but also gives insight into whether the original set of features is adequate to perform the classification task. If the set is not adequate, then feature extraction may be appropriate.

2.4.2 Feature Extraction

Feature selection can eliminate low quality features and produce a lower dimensional instance space. Feature selection risks losing information, however, and is not effective if none of the original features are good. Therefore, the creation of new features, called *feature extraction* or *constructive induction*, may be desirable. As with feature selection, considerable feature extraction is often done in choosing what observations to make of the environment, designing sensors or reporting procedures, and so on. Much of this work, for instance research on knowledge-based vision or extracting phonetic features in speech understanding, is very domain specific.

The literature on feature extraction is vast, particularly in applied statistics, where it is discussed under headings such as *transformations* ([Tuk57], [BC64], [Cha88], pp. 43-44; [Han81], chs. 5 & 6), *re-expressing the data* ([Tuk77], ch. 3), and *smoothing* ([Tuk77], chs. 7 & 16). We therefore survey below only some of the main issues and some important techniques in this area, concentrating on literature from the machine learning community.

2.4.2.1 Operators

All feature extraction methods require the use of some set of *operators* that can be applied to one or more current features to produce one or more new features. We can view an operator as a function that maps from sets of features to sets of features. Operators vary in their range of applicability. Logical operators such as conjunction and disjunction can be applied to all boolean features. Operators such as the arithmetic mean, multiplication, linear combination, and threshold functions can be sensibly applied to many numeric features. Operators can also be specific to a particular problem, such as operators for generalizing tic-tac-toe board features [Mat90].

2.4.2.2 Range and Domain of Operators

Operators can be classified according to the number of features they examine and the number of features they create.

A *one-to-one* operator maps a single existing feature into a single new feature. Transformations that replace feature values by roots or logarithms, such as are widely used in applied statistics, are examples of one-to-one operators.

One-to-many operators are unusual, though of course any set of one-to-one operators can be considered a one-to-many operator. A more interesting case of a one-to-many operator is the replacement of a k-valued nominal feature with k binary features, each corresponding to one of the original values ([Han81], p. 91). This sort of transformation is necessary to apply numeric-oriented learning algorithms, such as linear classifier builders, to nominal data.

Many-to-one operators are widely used, since they are a natural approach to dimensionality reduction. A widely used strategy is the application of conjunction or disjunction to boolean features. For numeric variables, linear combinations are a common approach [BFOS84, UB90]. When a set of original features are thought to be redundant manifestations of the same underlying feature, replacing them with a single feature corresponding to their sum, disjunction, mean, or some other cumulative operation is a good approach.

One widely used class of *many-to-many* operators, variously called *factor analysis*, *principal components analysis*, *canonical analysis*, and *discriminant analysis* produces a new set of features that is a linear transformation of the original features [Rum70, DH73, Bow84, Jam85, Kit86]. Another approach, which appears to have been used mostly in information retrieval, is to apply cluster analysis to the original feature set, resulting in a group of new features with less dependence among features [LC90]. Note that each of the individual features that result from a many-to-many operator is a feature that could have also been obtained by some many-to-one operator. The difference is that a many-to-many operator attempts to optimize some property of the entire output feature set, rather than a property of individual features.

2.4.2.3 When Are Operators Applied?

Operators may be applied at several points during the learning process. Many can be applied directly to definitions of features before the values of those features on any instances are known. Others, such as scaling a numeric feature to have mean 0 and variance 1 on the training set, require an initial analysis of the training corpus before application. One could even imagine operators that would, for instance, scale a numeric feature to have mean 0 and variance 1 on the test set, though this would restrict the classifier to acting on batches of instances.

Application of feature extraction operators can also be interleaved with learning, allowing characteristics of the classifier being induced to guide feature extraction. When feature extaction is interleaved with classifier formation it may be difficult to distinguish the two. For instance, training a connectionist network with hidden units can be viewed as parameter adjustment on a very complex functional form. On the other hand, the hidden units can be viewed as doing feature extraction [SW90], with a relatively simple classifier being trained to use these features.

2.4.2.4 *What Information is Used to Control Feature Extraction*

If the operators used are few and limited in their domain, then it may be practical to apply all operators to all applicable features. In most cases, however, applying all operators available to all features would be too expensive computationally and would lead to too many new features. Matheus and Rendell [MR89] list four approaches to controlling the application of feature extraction operators: *algorithm biases*, *training set biases*, *concept-based biases*, and *domain-based biases*. These correspond, respectively, making use of no information (random or arbitrary control), making use of properties of the training instances, making use of properties of the evolving classifier, and making use of domain knowledge.

This is a useful starting point, but we believe a finer-grained breakdown is desirable, as well as one that emphasizes what data is examined in deciding which features to construct:

1. Fixed control
2. Random control
3. Training set control
   (a) Unsupervised training set control
   (b) Supervised training set control
4. Classifier control
   (a) Intra-concept classifier control
      i. Structural intra-concept classifer control
      ii. Accuracy-based intra-concept classifier control
   (b) Inter-concept classifier control
      i. Structural inter-concept classifer control
      ii. Accuracy-based inter-concept classifier control
5. Knowledge-based control

Many systems will, of course, use a combination of these methods.

In *fixed control*, the algorithm applies feature extraction operators in some predefined order, which may be arbitrary or may have been chosen with expected characteristics of a class of problems in mind. In *random control*, there is a random component to which feature extraction operators are applied, usually with the assumption that a feature selection stage will choose the best of the resulting features. Research on genetic algorithms in machine learning [DeJ88] makes use of this approach.

Unsupervised training set control of feature extraction involves examining the values of predictor features on training instances, but not the values of criterion

features. For example, in using factor analysis to extract linear combinations of the original features, the extraction of features can be ordered by how much of the variance they explain in the training data. Canonical analysis is a *supervised* variant on this approach, extracting first those features that best distinguish the criterion classes on the training data ([Jam85], ch. 7).

There are a variety of ways to let the evolving classifier drive the construction of features. An example of using the structure of the evolving classifier is the FRINGE algorithm, which forms new features from pairs of features occurring near the leaves of a decision tree [Pag89]. Accuracy of the learned classifier can be used to guide feature formation when cross-validation (holding out part of the training set for evaluating classifier accuracy) is used ([BFOS84], p. 140) or in an incremental context [FU91].

If multiple classifiers are to be learned over the same data, either sequentially or simultaneously, the structure or accuracy of one classifier could be used to control operator application in learning other concepts. An example of the simultaneous case is the implicit formation of shared features at hidden units when training multi-output neural nets [DHB90]. The sequential case has been investigated in information retrieval, where learning of multiple concepts over the same database is the rule rather than exception (see Section 3.3.2.1). Both structural and accuracy-based methods are possible.

We reserve the term *knowledge-based control* to refer to the use of an explicit knowledge base separate from the learning algorithm to control the application of feature extraction operators. This can be distinguished from fixed control (which may have involved world knowledge on the part of the implementer) and the use of knowledge-based operators (discussed later in this chapter). As an example, the CITRE system makes use of knowledge-based control over the application of a knowledge-free operator, conjunction [MR89].

Control over feature generation can be exercised before the features are actually generated, based on known properties of the operators, or after feature generation by first generating features and then using feature selection to choose among generated features. Seshu, et al [SRT88] have argued for a strategy of *test incorporation*, where the evaluation of attribute quality is incorporated into the early stages of the feature extraction process, so that large sets of nonuseful attributes are ruled out as quickly as possible.

2.4.2.5 Replacement or Augmentation

If old features are retained when new ones are created, then dimensionality is increased, which is usually not desirable. Many-to-many operators, such as factor analysis, are used to replace the original feature set with an entirely new set. Often, however, it is desirable to use a mixture of old and new features. A common approach is to follow feature extraction by feature selection from the union of old and new features. Systems that incrementally create new features in response to classification effectiveness sometimes set a limit on the number of features, so that a new feature is accepted only if it can displace an old one [FU91].

Always keeping the best scoring features has the difficulty that a number of individually high scoring but redundant features may be selected. Stepwise feature selection methods can help with this. Alternately, when there are known relationships between features as, for instance, when one is constructed from another deductively [DR89, DCR89] it may be possible to substitute one feature for another in a controlled fashion.

2.4.2.6 What Information is Used in Constructing Features

The question of what information is used by operators in forming features is very similar to the question of what information is used in controlling operators. To some extent, the distinction between the two depends on the granularity at which one examines the learning system. A system like CITRE can be viewed as using domain knowledge to control the application of a simple operator (AND), or alternately can be viewed as having a complex knowledge-based operator whose output happens to end up being conjunctions. On the other hand, there does seem to be a difference between, say, favoring the use of AND over OR as an operator because many features have skewed distributions on the training corpus, versus creating a numeric feature that incorporates the sample mean of an original feature on the training corpus (see below). The latter clearly is information accessed by the operator, rather than by the mechanism applying the operator.

A similar breakdown of information sources as used in discussing control is appropriate:

1. Fixed operators
2. Random operators
3. Training-set-based operators
   (a) Unsupervised training-set-based operators
   (b) Supervised training-set-based operators
4. Classifier-based operators
5. Knowledge-based operators

We deemphasize the classifier-based case, as discussed below.

The simplest cases again are *fixed operators*, such as conjunction or multiplication, which make no use of additional knowledge in forming features. *Random operators* make use of some random component and their use is hard to distinguish from random application of nonrandom operators.

An important example of an operator that makes unsupervised use of the training set is:

$$g(x) = \frac{f(x) - \mu}{\sigma}$$

where μ is the mean value of original feature $f(x)$ on the training set, and σ is the standard deviation of $f(x)$ on the training set. The new feature, $g(x)$ has mean 0 and

variance 1 on the training set and, it is intended, approximates these characteristics on test sets ([JD88], p. 24). This and similar transformations are often used to keep features with large values or high variance from dominating the classification. Feature clustering and factor analysis are other examples of unsupervised feature extraction.

Supervised training-set-based operators are supervised learning mechanisms themselves, since they examine labeled training data and form a new function, in this case one intended to be used as a feature. The reason behind using an additional learning algorithm to form features is usually to compensate for too strong a hypothesis space bias in the primary learning algorithm, as in the use of linear discriminants at decision tree nodes [UB90].

Classifier-based operators examine not the labeled training data, but a classifier that has been formed from labeled training data. All the variations that were discussed under classifier-based control are available, and all are similar in that they examine some learned classifier, and typically make some subpart of it or modified subpart a new feature.

It is important to recognize that all the methods described above are limited, just as the main learning algorithm is, by the information available in the training data. Fawcett and Utgoff [FU91] would classify all of them as being *empirical* methods, to be constrasted with *analytical*, or *knowledge-based* methods. The use of domain knowledge is widely believed to be important in feature formation. Horn, discussing classification in computer vision, puts it this way ([Hor86], p. 344): "When you have difficulty in classification, do not look for ever more esoteric mathematical tricks; instead, find better features."

While domain knowledge is widely used in the original human selection of features for classification problems, only recently have methods been explored for automatically using a domain knowledge base during feature extraction. Domain-knowledge-based operators typically are transformations to be applied to statements in a logical language, such as propositional logic [DCR89] or predicate calculus [CU91]. They result in new features defined in terms of existing features by expressions in the appropriate language.

An alternative is to use a domain theory in a more limited way, to specify what initial features should be combined to form new features, while leaving it to some other mechanism to find the exact method of combining them. An example is research on knowledge-based neural networks [TSN90], where connections to hidden nodes are specified by a domain theory but backpropagation is used to weight these connections.

2.5 Summary

The preceding sections can be summarized in one sentence: Learning is hard, and there is a limited amount that can be done to make it easier without introducing new features. Quinlan presents an example where defining features for a classification problem required two man-months of effort, after which learning a perfect classifier was trivial [Qui83].

The importance of features has led to considerable research on what makes a set of features good, and on feature selection and feature extraction. A good deal is known, particularly about ways of addressing feature set inadequacies of a statistical nature, such as high dimensionality, noise, and redundancy. However, much remains to be learned, particularly with respect to the use of domain knowledge in feature extraction.

Having looked at issues of representation in the general case of classification, we turn to the issue of representation for the particular classification task of interest here, text classification.

CHAPTER 3

REPRESENTATION IN INFORMATION RETRIEVAL

In this chapter, we take an approach similar to that of Chapter 2, but focus exclusively on representation quality in information retrieval, and on the text representations used in text classification tasks. We first review some of the kinds of text representations used in IR, and clarify the relationship between indexing terms in IR and features in machine learning. We then survey theoretical research on representation quality in IR, particularly research on models that make predictions about representation quality. Following that we examine feature selection and feature extraction methods in IR, drawing parallels with research from machine learning. Finally, we discuss what conclusions can be drawn based on our survey of representation quality. In particular, we present the framework for a new theoretical model of text classification in which representation quality plays a more natural role. We also present a list of desirable properties of text representations.

3.1 A Review of Text Representations

In Chapter 1 we left the concept of a text representation rather vague. Having reviewed research on classification in Chapter 2, we can see that a text representation plays the same role in text classification systems that a feature set does for classification systems in general. We first discuss some of the terminology and ideas associated with features and feature values in IR, and then briefly consider the major kinds of text representations in use.

3.1.1 Terms and Term Weighting

A text representation or indexing language is a set of indexing terms. The phrase *indexing term* or just *term* is used in IR to refer to the same kind of entity as a feature in classification, that is, a function from instances (in this case documents) to some set of values.[1] We will refer to the *value* of a term for a document, just

[1] As in other classification tasks, there has been some use in text classification of representations that do not fit the feature set model [LCB89]. However, since there is little in the way of theoretical or experimental results about such representations for text classification, we will not treat them here.

as we refer to the value of feature for an instance. However, there are also ways in which terms are talked about in IR that vary considerably from how features are talked about in classification, and some discussion of this terminology is important to understanding the IR literature on text representation.

The form of a document that is initially available to a text classification system is typically a string of characters. To determine what the value of a term for a document is, the character string must be examined by some feature extraction process. For instance, the feature extraction process may consist of a human indexer reading the document and making an intellectual decision about which of a set of terms best captures the document's content. For binary terms, we say that a term is *present in* or *assigned to* a document to indicate that the term takes on the value 1 or *True* for that document. Conversely we say that the term is *not present in* or *not assigned to* the document if it takes on the value 0 or *False*.

One often encounters the notion of the number of occurrences of a term in a document, referred to as the *within document frequency (wdf)* or, somewhat ambiguously, as the *term frequency (tf)*. The within document frequency of a term refers to the number of separate occurrences in a document's character string of a linguistic clue indicating that the term should have a nondefault value for the document. One may talk about the within document frequency of a binary term, or may actually use the within document frequency as the term's value, in which case we have an integer-valued term. Numeric transformations of the within document frequency are also widely used, producing real-valued terms.

In discussing text retrieval, reference is often made to the *query terms*. These are the terms that constitute the feature set for a particular retrieval. Typically the query terms are only a small subset of the terms in the text representation, and are selected via some interaction with the user of the text retrieval system. This interaction often involves analysis of the natural language text of a request by the user. *Query expansion* refers to adding terms to the feature set for a retrieval, and may be done on the basis of interaction with the user or analysis of initially retrieved documents. When expansion is done using terms from retrieved documents, the new terms are called *feedback terms*. This form of query expansion is similar to feature selection mechanisms used in incremental machine learning algorithms.

In IR, *term weighting* refers to any mechanism that associates non-binary numeric values with terms. From the standpoint of classification, there are two very different mechanisms that are lumped together under the heading of term weighting. The first, called *term significance weights* [Cro81, Cro83], describes the importance of a term in representing a particular document. These are just feature values, and are sometimes referred to as the *document weights* of terms.

On the other hand, IR also uses weights derived from properties of the term in the whole collection or in a set of identified relevant and nonrelevant documents. These are sometimes called *query weights*. Mathematically, some of these weights can be viewed as initial parameter values for a classifier function, and others are parameters in parameterized transformations of the original term values. An example is the *inverse document frequency* or *idf* weight [Spa72]. If we let n_i be the number of documents in the text database containing term i, a common definition for the *idf* weight is $\log N/n_i$, where N is the total number of documents.

A widely used retrieval method is $tf \times idf$ weighting [SB87]. In this method a document j has a score found by taking the sum over all terms in the query of the product $tf_{ij} \times idf_i$ for each term i, usually with some normalization for document length. The tf weight for term i in document j will typically be some numeric transformation of the within document frequency weight, and usually will be 0 for many term/document pairs. In implementing $tf \times idf$ weighting, the product $tf \times idf$ is often stored rather than the term value tf. This increases efficiency at retrieval time, and is possible because idf values for terms stay constant as long as the collection of documents does not change.

There are at least two possible interpretations of $tf \times idf$ weighting from a classification viewpoint. The first is that tf weights are feature values, and idf weights are parameters to a classifier. The second is that tf weights are the original feature values, but that we transform the original features by multiplying by idf weights. The parameters to the classifier in this case are all 1.0. The situation is even less clear in variants of $tf \times idf$ weighting where idf values are used in computing both classifier weights and term values [SB87].

In the end, whether a weight should be considered a feature value, a transformation parameter, or a classifier parameter depends on the theoretical model from which the weights are derived. For instance, a Bayesian justificiation has been given for idf weights. In this model idf weights are parameter values of a classifier [Cro81]. The theoretical basis for using them in a multiplicative transformation of feature values is less clear.

The above discussion has assumed that however feature values are defined, they are known exactly. An important alternate view is probabilistic indexing, which is described in Section 9.2.2.3.

3.1.2 Major Text Representations

There are two major dimensions along which text representations are distinguished ([Sal89], p. 276):

- Indexing by humans vs. indexing by automated methods.
- Indexing based on the original text vs. indexing using a fixed set of terms.

From a classification standpoint, the four combinations of these methods are four different approaches to extracting features from the raw sequence of characters for the text. Automated indexing using terms drawn from the original text means that some automated feature extraction procedure is used. This typically involves isolating from documents those substrings or other linguistic clues likely to correspond to words or larger structures in a natural language. Each term corresponds to one such linguistic clue, and the set of terms corresponds to the set of all distinct clues encountered, under some definition of distinctness. The value of a term for a document typically depends on the number and position of occurrences of the linguistic clues in the document, as discussed above. One characteristic of automated

indexing from free text is that the set of features can grow as new documents are encountered.

The other widely used combination of methods is human assignment of terms from a fixed set, or *manual controlled vocabulary indexing*. Here the set of features is specified manually. Specification of the value of each feature for each document is also done manually. Usually these values are binary, or ordinal with grades such as *major/minor/not present*. As mentioned in Chapter 1, the effectiveness of controlled vocabulary indexing has been found to be quite similar to that of automated indexing from text.

Manual assignment of terms from free text is fairly widely used as well, though not as often as controlled vocabulary indexing, and usually as a supplement to controlled vocabulary indexing. Automated assignment of controlled vocabulary terms is simply the use of text categorization as a feature extraction strategy rather than as an end in itself, and in operational settings may be used either autonomously or as an aid for human indexers.

Within each of these four broad classes of representation there are substantial variations. Many of these variations can be viewed as resulting from the application of feature selection or feature extraction to simpler text representations. In addition, there are a variety of other representations that fit less well into the above model. Salton refers to information about documents such as author name, publisher, and citation data, as *objective identifiers*, since there is little debate about how to assign them to documents [Sal89]. However, these objective identifiers can also be used as indirect clues to content, and for this purpose can be treated much like other content features [FNL88].

3.2 The Quality of Text Representations

Experimental comparisons of text representation quality have been widespread in information retrieval. Most of these comparisons have applied a fixed retrieval method to the requests and documents of some IR test collection, and then varied the text representation used for documents. Almost all such experiments have been done using the text retrieval task, rather than other text classification tasks.

An experiment of this sort measures the relative quality of text representations given a particular retrieval method, a particular request set, and a particular document set. If the result holds up for other retrieval methods, other document sets, and other request sets then we can begin to have confidence in generalizations, such as those we listed in Chapter 1, about the text representations. If the same result holds up for a range of text representations sharing some characteristic, then we may be able to draw some conclusion about the relationship between that characteristic and text classification effectiveness.

Unfortunately, the characteristics of text representations measured in and controlled for in many of these experiments do not give us much insight into how text representations impact effectiveness or what the quality of proposed new text representations might be. This is particularly true for studies on manually produced representations, where the variation among manually defined sets of categories,

and the variation in the behavior of human indexers, is substantial and difficult to characterize. Other studies are hard to interpret because characteristics of the text representations are described only vaguely, or because of uncontrolled influences on effectiveness.

Nevertheless, some research has pursued connections between measurable characteristics of text representations and the effectiveness of these representations for text classification. We review this research below, first considering measures of the quality of individual indexing terms, and then measures of the quality of full text representations. We focus on research that presents theoretical models of representation quality that have the potential to predict the quality of new text representations.

3.2.1 What Makes a Good Indexing Term?

In this section we consider theoretical models that specify what characteristics a good indexing term should have, i.e. what makes a function from documents to values a good feature for text classification tasks. Two main types of models have been explored: those based on similarity measures between documents and those based on decision-theoretic cost estimates.

3.2.1.1 The Term Discrimination Model and Related Models

Salton, Yang, and Yu [SYY75] introduced the idea of the *discrimination value* of an indexing term. Their term discrimination model assumes documents are vectors of indexing term values, and that the similarity between documents is inversely proportional to the angle between these vectors. The discrimination value of an indexing term is the increase or to decrease in the mean inter-document distance caused by adding the indexing term to a text representation.

The term discrimination model assumes that increasing the average distance between documents will lead to better retrieval effectiveness, by allowing documents to be distinguished more easily from their neighbors. Therefore a term with a high discrimination value is a desirable indexing term. Salton, Wong, and Yang present examples of four feature set transformations where the increase or decrease in effectiveness has the relationship to average distance predicted by the term discrimination model [SWY75].

Empirical studies show that the discrimination value of an individual term is strongly correlated with the document frequency of a term, i.e. the number of documents for which the term takes on a nonzero value [SYY75]. Terms with the highest document frequency are the worst discriminators, but terms with very low frequency are also poor discriminators. Terms with document frequency of $n/100$ to $n/10$, where n is the number of documents in the collection, were found to have the highest discrimination values.

Based on this evidence, Salton [Sal86] suggests that high frequency terms be used as components in multi-term indexing phrases (which will have lower frequency than their parts), while low frequency terms should be grouped in thesaurus classes

(which will have higher frequencies than any of their individual members). Some experimental results have shown increases in retrieval effectiveness from the use of indexing phrases and clusters, though these improvements have been erratic.

Yu and Salton [YS77] have proven that certain transformations suggested by the term discrimination model will improve effectiveness. However, extensive assumptions are necessary for the proof, in particular the assumption that the query terms with higher document frequency occur in proportionally more nonrelevant documents, and that it is somehow known what terms to combine into thesaurus classes and phrases. Their analysis is also based on assuming that different term combinations are formed for each request. The proofs also apply only to vector space retrieval.

From the standpoint of giving guidance to the design of text representations for actual use, the predictions of the term discrimination model are weak. For instance, the model suggests that low frequency terms be combined to form thesaurus classes, but does not specify any other characteristics of terms to be combined. One is left to choose among a huge number of possible term combinations, most of them useless or harmful.

A variant on the term discrimination model, term precision weighting has also has been proven to improve effectiveness under certain assumptions, in this case assumptions more plausible than those described above. We discuss this method further in Section 3.3.2.

3.2.1.2 Probabilistic and Decision-Theoretic Models

Bookstein and Swanson [BS74] and Cooper [Coo78] have claimed that indexing should be done on decision-theoretic grounds by estimating the average cost across all users of indexing a document on a particular term. In other words, whether a particular index term should have a nondefault value for a document should depend on whether this is likely to increase or decrease the average cost to users of satisfying their information need.

Cooper suggested that this cost be estimated by thought experiments on the part of a human indexer, but this of course does not yield an automatic procedure. Bookstein, Swanson, and Harter [BS75, Har75b, Har75a] developed a utility-estimating model based on statistical assumptions about the distribution of terms. In particular, it assumes that a collection of documents is divided into two or more groups, each of which is about the concept specified by the indexing term to a differing degree. Terms are assumed to have values equal to their within document frequency. The distribution of these values within each group of documents is assumed to follow a Poisson distribution.

An approach to assigning terms to documents in a binary fashion has been developed for a special case of the above model where each term is characterized by exactly two Poisson distributions [Har75a]. However, this approach to computing term values has been found to be inferior to a simple binary model of term occurrence [Los88].

Maron [Mar79] develops a similar decision-theoretic model where each term is evaluated according to the probability that users mentioning that term in their

request will want a particular document. All terms for which this probability is above some threshold are assigned. No method for estimating the relevant probabilities is provided, however.

The Bookstein, Swanson, Harter model and the Maron model include a threshold that can be varied according to the cost of retrieving a nonrelevant document and the cost of missing a relevant document. Decreasing the threshold increases the probability that a term will be assigned to a document. Maron argues that this idea of a threshold should supersede the traditional notion of *exhaustivity* or *indexing depth*, which is usually operationalized as the average number of terms assigned to a document ([Tag81], p. 62).

A survey of decision-theoretic indexing models is presented in [CM78]. These models have had considerably more success from a standpoint of theoretical cleanliness than from one of text classification effectiveness.

3.2.2 What Makes a Good Set of Indexing Terms?

We turn in this section from characteristics of individual terms to characteristics of complete text representations (sets of indexing terms). Our focus is on attempts to find theoretical models connecting measurable characteristics of text representations to the effectiveness of text retrieval using those representations. We treat only briefly discussions of manual indexing of documents.

Svenonius [Sve86] gives a good review of what is known about controlled vocabularies and what questions are still open. She stresses linguistic justifications (synonymy and ambiguity of natural language words) as reasons why controlled vocabularies should work better. One strategy in controlled vocabulary indexing is to replace what would be synonymous terms in a word-based representation by a single identifier in the controlled vocabulary. Svenonius suggests that control for synonymy may have different effects for different collections, based on Bhattacharyya's measure of the average *terminological consistency* [Bha74] of the discipline from which the text is drawn. This value is defined as:

$$T = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{s_i}$$

where n is the number of "concept terms" in the field, and s_i is the number of synonyms of the ith concept term. However, since there is no widely accepted definition of what a concept term is, or even of what a synonym is, this measure is not important helpful. The only clear finding Svenonius claims for previous research on controlled vocabulary indexing is that different representations tend to retrieve different documents [KMT+82]. This phenomenon has been used to increase text retrieval effectiveness by combining multiple text representations [CLC88, FNL88, Tur90].

Svenonius mentions, but expresses some doubt in, the theoretical and empirical result that increasing the exhaustivity of indexing tends to increase recall and decrease precision. Sparck Jones [Spa81], among others, draws this conclusion from previous work, but has also pointed out that degree of exhaustivity can be

compensated for in query formation [Spa73b]. This is under the traditional definition of exhaustivity, i.e. the number of terms assigned per document. This traditional measure actually confounds three text representation characteristics: dimensionality (number of terms in the text representation), the set of concepts represented in the vocabulary (in particular how specific or general they are), and an indexing threshold of the type discussed by Maron and Harter. Only the first two characteristics are inherent to the text representation.

Other research on the quality of term sets has come not from attempting to formulate satisfactory policies for manual indexing, but instead from trying to understand why existing representations exhibit good or bad effectiveness. The best example of this research was Sparck Jones' attempt to explain why term clustering led to improved effectiveness on one test collection and not on two others [Spa73a]. She proposed, tested, and rejected a number of hypotheses before adapting the test Van Rijsbergen proposed for the Cluster Hypothesis [Jv71, van72]. This test compares the distribution of a similarity measure over two groups of document pairs. One group contains all pairwise combinations of relevant documents, while the other contains all pairs consisting of one relevant document and one nonrelevant document. The more the distributions overlap, the worse the separation between relevant and nonrelevant documents. For the collections that Sparck Jones was studying, a much larger overlap was found for the two poorly performing collections than for the well-performing collection.

This led Van Rijsbergen and Sparck Jones [vS73] to propose the use of this Cluster Hypothesis Test (CHT) as a way of determining whether an initial set of terms will be improved by methods such as term clustering, which are meant to drive apart relevant and nonrelevant documents. They also suggest that the test can be used to determine whether a new indexing vocabulary for a collection will actually lead to effectiveness improvements. The CHT was later used by Burnett, et al [BCL+79] to compare a number of indexing vocabularies of various sizes, and the results generally showed that representation quality as predicted by the test correlated with retrieval effectiveness. The CHT has so far been used only in a heuristic fashion. Further tests of its efficacy, and a theoretical explanation for its usefulness, would be desirable.

Another measure, also proposed in terms of evaluating collections and also applicable as a measure of representation quality, is Sparck Jones's "performance yardstick" [Spa75]. This involves finding the recall-precision curve achieved by weighting query terms according to their distribution in the complete set of relevant and nonrelevant documents. What this really means is that one forms the optimal classifier possible within the limits of a given machine learning algorithm, and testing on the training data. Effectiveness is usually not perfect since only query terms derived from typically short user requests are used as features.

Wong, Yao, and Bollman [WYB88] applied a similar strategy, using a feature set including all terms in a text representation, to make claims about the adequacy of linear classifiers in IR. They used gradient descent on the perceptron criterion function to train perfect or near-perfect linear classifiers for all requests in two collections. However, this tells us nothing about the representations used except

that their dimensionality was large in comparison to the number of documents. The same can be said for the small, optimal boolean queries found by Gifford and Baumanis [GB69].

Another view on the nature of good term sets comes from Brookes' information-theoretic analysis of IR [Bro72]. Under the assumption that we want to maximize the information transmitted per term assigned, given a fixed total number of term assignments, then we want each term to be assigned to the same number of documents. Since free-text terms actually follow a Zipf's law distribution, this would suggest that considerable improvements are possible by representation changes, such as those suggested by Salton on term discrimination grounds, that tend to equalize the frequency of terms. On the other hand, Brookes points out that since terms will undoubtedly not be distributed equifrequently in user queries, achieving equal information content per term may not be desirable. Brookes does not investigate how one would derive the ideal distribution of document terms given a known distribution of query terms.

3.3 Improving Text Representations

Improving of text representations has been an area of considerable interest in information retrieval. We will see that many of the same issues are encountered as in feature selection and feature extraction for machine learning in general. However, the specific characteristics of text classification tasks have led to some novel approaches.

3.3.1 Feature Selection for Text Representation

Feature selection mechanisms are important in classification problems when the number of potential features is large in comparison to the number of training instances, or when features have any of a variety of problems with quality. In this section we discuss feature selection for text retrieval and for text categorization.

3.3.1.1 Feature Selection for Text Retrieval

Text retrieval is an extreme case of the need for feature selection, since there are often tens of thousands of features (indexing terms) and few or, in the initial retrieval, no training examples available. This means that information other than training instances must be used in feature selection.

One approach to feature selection for the initial retrieval is to use the user request. The usual approach is to apply the same language-processing mechanism used to determine feature values for documents to the text of the user's natural language request. The set of features that would be given nondefault values if the request was a document is used as the initial text representation for that retrieval. This is the approach taken in probabilistic retrieval methods [CH79].

Another approach is to form a classifier that is less affected by a high dimensionality feature set. This is the case in vector space retrieval [SM83]. Again the

user request is processed in a fashion similar to that for documents, and serves as a prototype against which other documents are compared. The cosine correlation is used to measure how similar each document is to the prototype, and this similarity value is the system's estimate of a document's membership in the class of relevant documents. Feature selection is avoided (or perhaps should be viewed as implicit) by setting the value of all features that were not detected in the request to 0. Such features, no matter how many there are, can have no effect on the classifier output, under the cosine similarity formula.

Other sources of information can be used to eliminate certain features before any user requests are processed. One common strategy is a *stop list*—a fixed set of words that are never used to index documents. The justification for using such a list is usually asserted to be both linguistic and statistical. Van Rijsbergen ([van79], p. 17) talks about "the removal of high frequency words, 'stop words', or 'fluff' words", Salton and McGill ([SM83], p. 71) call them the "high-frequency function words", and Vickery and Vickery ([VV87], p. 122) "very frequent non-significant words." There is some contradiction in these views, since some words that are function words have low frequency, and some very high frequency words are not function words. This includes some of the words contained in typical stop lists, such as Van Rijsbergen's ([van79] , pp. 18–19). However, we can view stop lists as primarily being defined linguistically, and they are perhaps the only solidly successful application of linguistic knowledge to improving representation quality in IR.

The problem of feature selection becomes more traditional, but also more acute, in a relevance feedback context. Since the user has been shown a small number of documents and has indicated whether each is relevant or non-relevant, there is a small training set available. This training set can be used for feature selection, as well as for adjusting parameters of the classifier. (Note that before relevance feedback, default or user-supplied values must be used for these parameters.)

The small number of training examples available during relevance feedback affects feature selection as well as classifier formation. There is considerable controversy over the degree to which feature selection should be done during relevance feedback. Some authors suggest expanding the query with all features that have non-default values in documents judged to be relevant [SK86, SB90], while others suggest more selective automatic methods or relying on user judgment [Hv78, vHP81, Sv83, Har88b, CD90]. Results in direct comparisons have been mixed [SB90, CD90]. Still another approach is to leave the set of features suggested by the user request unchanged, and use the feedback documents only for tuning the classifier parameters [Spa79, WS81]. This latter approach is called *relevance weighting.*

3.3.1.2 Feature Selection for Text Categorization

Feature selection in a text categorization context is more like the usual machine learning case than is feature selection in text retrieval. A large number of classified examples are often available. Despite this, relatively few of the automated feature selection techniques explored in machine learning have been tried for this task. A

few measures of association between term occurrences and category occurrences have been used or suggested [Mar61, Fie75, HZ80, BFL+88]. For the most part, however, manual feature selection has been relied on, along with stop lists as described above. Given that the classifiers for a text categorization system can be expected to be in use over a long period of time, the investment of human effort in feature selection is not unreasonable, though it seems likely that automated aid would both save effort and increase effectiveness.

3.3.2 Feature Extraction for Text Representation

As with feature selection, feature extraction has been the subject of considerable research in information retrieval, again almost solely for text retrieval rather than text categorization. We separate our discussion here into methods that have been or could be used outside of text classification, and those that are specific to text classification problems.

3.3.2.1 Domain-Independent Feature Extraction in IR

Many of the domain independent feature extraction mechanisms described in Chapter 2, particularly the ones developed in statistics, have been applied in the IR context. One-to-one transformations, such as taking logarithms, are often used alone or as part of term weighting formulas. One-to-many operators are rare, however, as in most other classification tasks.

The primary many-to-one operator used is *indexing phrase formation.*[2] An indexing phrase is an indexing term that corresponds to the presence of two or more single word indexing terms. Many phrase formation techniques used are specific to the IR domain, since they are based on relationships between words in text. However, one method of phrase formation is simply to take the conjunction of two or more existing terms, a method widely used in machine learning.

Many-to-many transformations applied to IR have included feature clustering (term clustering), pattern clustering (document clustering), and factor analysis (latent indexing). As with related techniques in machine learning, these domain-independent feature extraction strategies usually have yielded only small effectiveness improvements.

Text retrieval is unusual among classification tasks in that it is necessary to induce large numbers of classifiers over the same set of data. Each time a user makes use of the text retrieval system, one or more new classifiers is created. This has led IR researchers to investigate a variety of methods for intra-concept classifier control in feature extraction.

A number of researchers have investigated *document space modification*, i.e. changing the value that each term has for a particular document, based on whether

[2]In the IR literature, the term "phrase", rather than the longer "indexing phrase" is usually used. We will usually use the longer term, as "phrase" is widely used with other meanings in natural language processing and linguistics.

or not that document should have been classified as relevant by classifiers using that term in their feature sets [FMW71, Bra71, Gor88a, Gor88b]. This is an unusual type of feature extraction, in that the value of the new feature for a document will depend on when the document entered the database, the requests for which it was retrieved, and to which of those requests it was judged relevant.

A drawback of document space modification methods is that term values change only for documents that have been the subject of relevance judgments, and then only for those terms that appeared in the query or the document. An approach that changes the value of a term for all documents containing the linguistic clue for the term is *term precision weighting*. In this approach, a sample of queries and relevance judgments is used to estimate the tendency of a term to occur in relevant vs. nonrelevant documents. Terms with a greater tendency to appear in relevant documents are weighted more highly for future queries. Yu and Salton have proven, under certain assumptions, that term precision weighting will improve retrieval effectiveness [YS76] and have demonstrated, under conditions somewhat different from those of the theoretical case, small improvements in effectiveness from this weighting [SWW76].

Fuhr and Buckley have recently demonstrated a method that allows transformation of all feature values for all documents [FB90]. The method assumes that several different ways of determining the value of a feature are available (such as the *idf* weight, the *wdf* weight, document length, and so on). The method trains the parameters of a polynominal combination of these different values. Even though only a subset of features and a subset of documents are seen during training, sufficient information is gathered about appropriate feature value combinations to show substantially increased effectiveness on new classifiers.

Traditional many-to-many feature extraction methods can also be guided by relevance judgments on previous user requests. This has been attempted with both term clustering [YR77] and document clustering [Wor71].

It should be pointed out that all studies in the area of intra-classifier feature extraction in IR have suffered from the relatively small number of requests, documents, and relevance judgments available in standard test collections.

3.3.2.2 Domain-Dependent Feature Extraction in IR

A variety of domain specific feature extraction methods have been developed in information retrieval. The four main approaches to text representation described earlier are simply four broad classes of feature extraction techniques, within which a number of variations have been tried.

Since documents are instances of human language, research on feature extraction for text classification tasks often draws explicitly or implicitly on notions of linguistic structure. Some of the linguistic assumptions made are so basic as to be almost unnoticable. For instance, documents are often discussed as if they were sequences of words, with the assumption that there is a single, straightforward technique for creating a text representation whose terms correspond to words. The form in which documents are usually made available to the computer is actually a string of

characters, however, and there are a number of significant decisions made in deciding how to segment that string into words. This is particularly true in languages such as Japanese, where the writing system does little to indicate word boundaries. Even in English, many unclear cases arise, particularly with respect to compound words, names, and hyphenated words. It is not even necessary to have a notion of words at all to derive a reasonable text representation, since indexing on short sequences of characters provides a representation with effectiveness not much worse than that of indexing on words [Wil79].

Nevertheless, most work on text representation has assumed a set of basic features corresponding to words. The most widely used feature extraction method applied to a word-based feature set is *stemming* ([van79], pp. 17–22; [SM83], pp. 71–73). Stemming is a knowledge-based feature clustering procedure. It applies heuristics for removing affixes from the word forms corresponding to terms, and forms clusters from terms that are reduced to the same form by this process. A term corresponding to the group of related terms is formed, and this term's value for a document is typically the disjunction or the sum of the values of the terms in the cluster. So, for instance, the terms corresponding to the words *computer*, *compute*, *computing*, and *computational* might be collapsed to form a single term, which might be denoted by the form *comput-*. Often this clustering process is incorporated into the procedure that produces the original text representation from the character strings for documents, so that terms corresponding to unstemmed words are never actually formed. A stemmed representation has been widely, but not uniformly [Har87], found to be more effective than an unstemmed one.

Another approach to knowledge-based feature extraction is to focus on the meaning of words. *Thesauri* are knowledge sources that specify semantic relationships between words. One approach to feature extraction using thesauri is to replace single word terms with clusters of terms specified as being related by a thesaurus. Another is to specify that the value of a term for a document takes on a nondefault value not only when the linguistic clue for that term is present, but also when a linguistic clue for a related term is present. Some experiments on thesaurus-based feature extraction have shown improvements in retrieval effectiveness [Fox80, WV85], but this technique is still poorly understood.

Any approach to text representation based on the meaning of words must address the dual problems of *ambiguity* (the fact that words can have several meanings) and *synonymy* (the fact that several words can have the same or closely related meanings). To address these problems, Krovetz has suggested that indexing terms correspond to word senses from a machine-readable dictionary rather than correspond to words [KC89].

Other text representation techniques have attempted to form indexing terms that correspond to linguistic structures larger than a word. A variety of techniques for multi-word indexing or *phrasal indexing* have been investigated. An approach that makes some use of document structure is *proximity phrase indexing*, where the phrasal term has its maximum value for a document only when linguistic clues for the words in a phrase are found within a certain distance of each other in the document. Frequency information is often used to decide which words to combine in

a proximity phrase, so these phrases are sometimes referred to as *statistical phrases.* A limiting case of proximity phrases is when the distance between linguistic clues for the component words is ignored, and we have the conjunction or sum of two terms, as discussed in the previous section.

A number of researchers have investigated *syntactic phrase indexing*, in which a phrase takes on its maximum value for a document only if the component words are in a specified syntactic relationship. Syntactic phrase indexing is discussed at length in Chapter 4. Surveys of approaches to both syntactic and statistical phrase indexing have appeared elsewhere [Fag87, CTL91].

We have not dealt here with manual indexing, since properties of this method are so affected by human variability. The Svenonius survey provides good pointers into this area [Sve86].

3.3.3 Contrasts with Machine Learning

Comparing work on feature set improvement in text classification with work in the more general field of machine learning shows many similar techniques have been applied. This is particularly true for methods that both fields have drawn from statistics, such as transformations, clustering, and factor analysis. Information-theoretic techniques for evaluating and selecting features have also been widely used by both fields.

However, text classification has also investigated methods rarely examined in machine learning. The emphasis on human feature selection, via the text of a user request or other interaction with the IR system, is the most notable case. Buntine [Bun90] and others have argued for just such an interactive approach to induction in the more general machine learning context. Another important area explored in information retrieval, but rarely in machine learning, is taking advantage of multiple concepts learned over the same set of instances. Finally, as in any classification application area, there are techniques specific to the domain. In the case of IR, these include manual indexing and various natural language processing methods.

Conversely, text classification research has taken little advantage of the more advanced techniques for feature extraction developed in machine learning. Techniques that make use of domain knowledge in some form may be of particular interest in counteracting the role of small training set size in relevance feedback [Lew91b]. On the whole, the methods used for building classifiers in text classification systems have been simple in comparison to recent methods developed in machine learning. Whether these more complex methods, which often interleave feature extraction and classifier formation, will be more effective remains to be seen.

Information retrieval and machine learning diverge from each other the most in their theoretical and heuristic attempts to understand the quality of feature sets. This is not surprising, since most research into text representation quality in information retrieval has focused on the text retrieval task, where the influences of human query formation, multiple queries, and, sometimes, manual indexing are predominant. Machine learning research on feature set quality, on the other hand, has focused on a traditional learning from examples paradigm. The closest

overlap is in techniques that are based on known groups of relevant and nonrelevant documents. For instance, the Cluster Hypothesis Test is similar to techniques for evaluating feature sets based on overlap of probability density functions [Kit86].

3.4 Discussion: Towards a Theory of Text Representation

Boyce and Kraft, in a survey of theories and models in IR [BK85] conclude :

> What we know about the characterization of documents on a level beyond practical rules of thumb is, however, not extensive.

and

> One may use probability theory, utility theory, the theory of syntax, or information theory to gain insight into the process of representing documents, but no such approach gives us a general theory of document representation.

On the other hand, it seems clear that representation is important. Gordon puts it this way [Gor88a]:

> Document retrieval is primarily a problem of representation: representing documents by storing some form of description of them in a database; and representing an inquirer's information needs with queries that can be processed by computer. If forming adequate representations of both documents and users' needs were completely understood, there would be no need for further research in this field.

Clearly there is still a need for further research in this field. However, the beginnings of an understanding of text representation can be drawn from the research surveyed in this and the previous chapter.

In this section we present two results of our examination of the role of representation in classification and text classification. The first is a new model of IR systems that emphasizes the role of classification and provides an answer to the Perfect Query Paradox. The second is a list of desirable properties of text representations, culled from the machine learning and IR literature, and motivated by our emphasis on the machine learning aspects of text classification.

3.4.1 The Concept Learning Model

The similarities of work on representation in text classification to that in the general field of machine learning are striking. Both fields have devoted considerable effort to addressing the difficulties for classification caused by the high dimensionality and low quality of feature sets. Some of the techniques used have been the same, and some have been different. Considerably more effort has been devoted in

machine learning than in IR to modeling the effect of representation on classifier effectiveness. The potential exists to draw on this work in developing theories of representation for IR.

Before this can be done, however, we must consider one substantial difference between text classification and most other classification tasks—the role of human beings in constructing classifiers. The influence of manual construction in text retrieval and text filtering is always substantial, and it can be substantial in text categorization too. The argument could be made that results on representation from machine learning are not relevant to text classification, since most text classification procedures involve a human component in addition to computer induction of classifiers.

One approach to this problem would be to look to machine learning for advice only in those cases where fully automated text classification is done, as in some text categorization systems. Some other approach would be required to deal with, for instance, text retrieval, where the role of the initial user request is so important.

An alternative is available, however, that allows us to apply more broadly results and insights from machine learning. Recall the Perfect Query Paradox introduced in Chapter 1: almost all current text representations allow perfect text retrieval with almost all query forms in use, but actual text retrieval effectiveness is far from perfect. Current models of text retrieval may explain the effectiveness of those components of an IR system that operate purely by machine learning, but they have nothing to say about the initial user request.

Suppose, however, that we step back and consider as a whole the IR system and the human user or, in the case of categorization systems, the human knowledge engineer. The result of their interaction at any point in time is a query, or profile, or categorizer than can be applied to documents to make a classification decision. Let us also make the (admittedly controversial) assumption that a human being can be usefully modeled as a finite computational device. *Then the combination of IR system and human is a finite computational device that examines examples and outputs a classifier.* It is a machine learning system.

We call this view of IR systems the *concept learning (CL) model.* Under this model the Perfect Query Paradox goes away. Like any machine learning system, the composite IR system will have some bias in its search of the space of possible classifiers. This bias may be such that it does not find the ideal classifier, or even a very good classifier. The bias of the composite system will result from the interaction of the biases of the human and mechanical components. In the CL model it is meaningful to talk about the effect of text representations on the composite bias of human and machine learning algorithm. This is not the case under IR models that cover only the machine learning component. The CL model also covers those cases where a text classifier is produced by purely automated means.

The CL model gives less insight into the Equal Effectiveness Paradox. It does suggest that we look for the answer in the interaction between the kinds of classes desired by users of text retrieval systems, and the kinds of classes defined by terms in common text representations. It is worth noting that both natural languages and controlled vocabularies are languages evolving from human populations.

As presented above the CL model is a mere skeleton. For the CL model to make strong predictions, it must be accompanied by models of the bias of the machine and human components, the latter of course being more difficult. Considerable research has been done on human limitations in request formulation for text retrieval systems [Fid91] and human biases in category formation [SM81], so useful models are not an impossible proposition.

For our current purposes the CL model as presented above suffices. It allows room for the possibility that one text representation can be better than another, and suggests that properties of representations found important in machine learning may be important in the special case of information retrieval.

3.4.2 Desirable Properties of Text Representations

In the following we provide a list of characteristics of representations found to impact effectiveness in previous research on classification or text classification. We have attempted to choose those characteristics that are supported by several threads of research, and to define the characteristics so that they are operational, i.e. can actually be assessed, at least qualitatively, for a representation. We also try to indicate some of the open questions associated with these measures.

1. *Small number of indexing terms.*

It is a theoretical and empiricial result in machine learning that, except in certain ideal conditions, there is a maximum number of features beyond which effectiveness of an induced classifier will begin to decline, given a fixed number of training instances. This "curse of dimensionality" holds even for induction methods that have the ability to downweight poor features or that incorporate feature selection. A crucial point to remember is that statistical feature selection methods are themselves inductive and therefore also are affected by large numbers of features.

The only case where the dimensionality effect has been directly tested in information retrieval is in query expansion during relevance feedback. Results here are currently controversial, with some authors arguing that all terms from relevant documents should be used for expansion (i.e. dimensionality is not a significant problem), and others arguing that only selected terms should be used (i.e. dimensionality is a significant problem, and feature selection is necessary). Experimental results are available supporting both positions, and the issue is complicated by the presence of terms from the original user request, and by the fact that training is done from a small, but high quality retrieved set. Taking the larger view, however, no researcher suggests that all terms in the entire text representation be used for feedback, something that is logically possible once a feedback set is available, and few suggest including terms from documents judged to be nonrelevant. To that extent all agree that dimensionality is an issue.

The problem of high dimensionality is more serious in the automated induction of classifiers for text categorization, since there is no user request as the default source for feature selection. While dimensionality effects are rarely discussed explicitly in the literature on text categorization, these effects are implicitly acknowledged by the widespread use of manual and automated feature selection in these systems.

2. *Flat distribution of values for an indexing term.*

For a feature taking on discrete values, it is desirable that the prior probabilities for these values be close to equal. For a feature taking on continuous values, the cumulative distribution function should be close to linear in the range where the feature takes on its values.

One theoretical argument for this characteristic comes from information-theoretic models, which indicate that a feature transmits maximum information when equiprobability of values holds [Bro72]. Of course, we are less concerned with the absolute information content of a feature than with the information it carries about the class or classes for which we want to learn classifiers.

Another argument is based on assuming that, implicitly or explicitly, the machine learning procedure used is based on estimates of the probability that a feature will take on particular values under particular conditions. For instance, it is often desirable to estimate the a priori probability that a feature takes on a particular value for an instance. If that value is taken on for very few members of a training corpus, then the estimate of the a priori probability is likely to be inaccurate [CG91].

Empirical results provide mixed evidence for the desirability of a flat distribution of feature values. Omission of high frequency terms is often found to improve effectiveness. Empirical results also show that medium frequency terms are the best discriminators under the assumptions of Salton's term discrimination model. However, the significance of these results is clouded by the fact that high frequency terms are often just plain bad content indicators, particularly when terms correspond to words from natural language.

To some extent the problems of skewed feature value distributions can be overcome by the use of domain knowledge. The reason that low frequency terms are found to be valuable in text retrieval experiments is probably that human beings rely on domain knowledge rather than or in addition to statistical properties when selecting such terms to include in requests. It is likely that low frequency terms are less useful in query expansion than when present in initial requests, but we do not know of experiments that have addressed this issue directly. A proper comparison would require controlling for the quality of the terms as content indicators. One approach might be to split high frequency terms randomly into artificial low frequency terms.

3. *Lack of redundancy among terms.*

We use the term "redundancy" to cover several problems whose influence is hard to tease apart. One is the notion that it is undesirable to have multiple indexing terms corresponding to the same underlying property of an object or to a closely related property. This results in counting the same piece of evidence more than once in deciding to which class a document should belong. Of course, defining notions such as "same underlying property" and "same piece of evidence" outside of artificial situations is quite difficult. For indexing terms that are derived from natural language text the linguistic notion of synonymy is clearly relevant here, but

we must rely on human judgment with respect to just what linguistic structures are synonymous.

A closely related issue is statistical dependence. Two events, A and B, are statistically independent if $P(AB) = P(A) \times P(B)$. A number of machine learning models, including most of those used in IR, explicitly or implicitly assume that terms take on values independently of each other under certain conditions. Also, from an information-theoretic viewpoint, the amount of information conveyed by a set of dependent features is less than the amount of information conveyed by a set of independent features.

Attempts to develop statistical text classification methods that specifically take into account term dependence have been hampered by the small number of training examples available in a relevance feedback context. Harper and Van Rijsbergen have shown that a limited dependence model improved effectiveness over an independence model in a *retrospective retrieval* test, where the entire collection is used for training parameters [Hv78]. This involves training on test data, so the results are useful only as upper bounds. For the standard retrieval environment they resorted to an independence model, using dependence information only for query expansion. More success has been obtained by taking advantage of human-provided information about term dependencies, and using this information in a nonlinear classifier [SFW83, Tur90, CTL91].

The relationship between statistical dependence and synonymy is complex and will not be explored here. The possibility of noise, and the range of possible definitions of underlying meaning, rule out any simple or uniform relationship between these properties. All else being equal, however, synonymy would tend to imply some degree of statistical dependence.

4. *Low noise in indexing term values.*

The concept of noise is elusive in classification in general, and is no less so in text classification. A term can only be said to be noisy with respect to some model of what its correct values should be, something that is rarely available for terms in text classification. Relatively unambiguous cases of noise are those introduced into word-based terms by spelling errors, optical character recognition, or speech recognition. It is much more difficult to say what it means for a term corresponding to a correctly recognized word to be noisy.

Even when we cannot specifically detect noise in a text representation, it can still be a useful concept, however. Combining redundant features can reduce whatever noise is present, and if no noise is present we are no worse off. So the possible presence of noise is another argument for combining redundant terms.

5. *Lack of ambiguity for linguistically derived terms.*

The previous four characteristics are present in the general classification literature as well as the text classification literature. Ambiguity, however, is a characteristic particular to human language. Ambiguity is hard to define explicitly beyond the rather vague notion of a word or other linguistic structure having more than

one meaning. From a text classification standpoint, an ambiguous word might be viewed as a disjunction over two or more unrelated subconcepts. Ambiguity is an undesirable characteristic in a text representation because random collections of unrelated subconcepts are unlikely to be combined in the class or classes to be distinguished by the text classification system.

Ambiguity has long been recognized as a problem in information retrieval. Direct approaches, i.e. attempting to define linguistic criteria for recognizing which sense of a word is being used are receiving increasing interest [KC89]. Indirect approaches to dealing with ambiguity included both term clustering [Spa71] and phrase formation (see Chapter 4).

6. *Terms should be related to the classes to be induced.*

This is just another way of saying, as we did in Chapter 2, that we would like a good set of features, a set of features that makes it easy to learn classifiers for the classes of interest. In text classification, as in other classification tasks, a great deal remains to be learned about what makes a good set of features.

There are a few nontrivial things that can be said about good terms. One is that in most text classification tasks we are interested in learning multiple classifiers over the same feature set and data. The quality of a set of terms must be considered with respect to a set of classes (as in many text categorization systems) or a distribution of possible classes (as in a text retrieval system), rather than with respect to a single class. Note that there is a tension between the desire to have a small feature set and the desire to have a feature set that contains good features for a wide variety of user information needs.

Another point is that many text classification systems include a human component, such as the user who produces a request to a text retrieval system. This raises a host of issues about the ability of humans to select terms from text representations. We have not attempted to deal with these issues here, but they are undoubtedly significant.

It is possible that the relative quality of text representations for the same set of documents may vary with the particular text classification task being performed. For instance, user needs in a text retrieval environment tend to be narrower in scope than, for instance, the categories used in a text categorization environment. Indexing terms with meanings whose breadth is comparable to that of single words might therefore be more appropriate for text categorization, while phrasal terms might be more appropriate for text retrieval. The other factors we have described would of course interfere with any simple relationship of this sort.

3.5 Summary

This chapter has surveyed research on text representation in information retrieval. Many of the techniques for representation quality improvement used in machine learning have also been investigated for text classification. However, there are techniques on both sides that have not been investigated by the other, so there are undoubtedly many profitable interactions that have not yet been explored.

A number of theoretical models of representation quality for IR have been developed. With the possible exception of the term discrimination model, they have had little impact on IR practice. The guidance provided to a representation designer by the term discrimination model is quite weak, however, and the representation improvement strategies suggested by it have had mixed results.

The applicability of results from machine learning can be questioned on the grounds of the substantial human role in classifier formation for many text classification tasks. We propose the concept learning model of text classification systems. The central tenet of this model is that both the human user or knowledge engineer, and the text classification system, have some bias in searching the space of possible classifiers. Understanding the quality of text representations will require understanding both of these biases. However, just making the assumption that the human bias is not perfect enables us to talk meaningfully about the relative quality of representations that would have to be considered equally effective under a perfect human bias.

Another result from our survey was a list of desirable characteristics of text representations, supported by theoretical and empirical work in both machine learning and text classification. These characteristics can provide guidance in constructing new text representations and understanding existing ones. We make just such an attempt in the next chapter with respect to a particularly difficult question in text representation, that of syntactic phrase indexing.

CHAPTER 4

SYNTACTIC PHRASE INDEXING

Our survey of previous research on representation led to a new model of text classification systems in which the role of text representation can be better understood. For the CL model to be useful, it should make testable predictions about the properties of text representations. Ideally, it should also lead to strategies for changing or replacing those representations so as to improve the effectiveness of text classification systems.

In this chapter, we examine a particular text representation, syntactic indexing phrases, from the standpoint of the CL model. Syntactic phrase indexing has been viewed as having great potential, but in text retrieval experiments has never led to significant effectiveness improvements. We provide an explanation for these results in terms of properties of a syntactic indexing phrase representation. We then suggest why a particular representation change technique, term clustering, might improve the properties of this representation.

We end the chapter by presenting the hypothesis that term clustering of syntactic indexing phrases will improve the effectiveness of a text classification system. This is the hypothesis that was explored in the research reported in the remaining chapters of this dissertation.

4.1 Research on Syntactic Phrase Indexing

By syntactic phrase indexing, we mean the application of analyses of the syntactic structure of natural language text to produce multi-word indexing terms. This was one of the earliest artificial intelligence techniques to be applied to IR, and has been one of the most widely investigated. The interest in this technique stems from the belief that many text retrieval errors are the result of representing documents at the level of individual words. Clearly the relationships among words in a piece of text are important to a human reader, and if some of these relationships were made available to an IR system, increased effectiveness might result.

The use of syntactic information for phrasal indexing has been surveyed elsewhere [Fag87, Sv88, LCB89], so we will keep our survey brief, emphasizing experimental comparisons of text retrieval effectiveness with and without syntactic phrase indexing. We divide the techniques investigated into two major classes: template-based and grammar-based.

Dillon and Gray's FASIT system [DG83] is typical of template-based phrasal indexers. Adjacent groups of words from documents are matched against a library of templates, such as <JJ-NN NN> (adjective noun), and <NN PP NN> (noun preposition noun), and those matching some template are retained. Most templates in FASIT and other template-based systems are oriented toward finding contiguous sequences of words that represent noun phrases. Phrases are normalized by stemming and removal of function words. Klingbiel's MAI system used a similar strategy [Kli73], while the TMC Indexer [MWRR86] and LEADER [HK69] combined limited parsing with templates. Most of these studies did not present effectiveness results. Dillon and Gray did compare FASIT's phrasal indexing with a conventional word-based indexing. Average precision of FASIT was found to be superior to that of word-based indexing, but only by a few percentage points and not at all recall levels.

Grammar-based strategies attempt to analyze entire sentences or significant parts of sentences in producing syntactic indexing phrases. Fagan [Fag87], for example, used the PLNLP parser to parse completely the text of two test collections and extract indexing phrases. We summarize Fagan's procedure in Figure 4.1. The sophistication of the PLNLP grammar enabled Fagan to handle complex noun phrases with prepositional and clausal postmodifiers, as well as some adjectival constructions. Fagan also used a number of hand-built exclusion lists of words which signaled that a phrase should not be generated, or should be generated in a special fashion.

On two test collections Fagan's syntactic indexing phrases produced improvements of 1.2% and 8.7% in average precision over indexing on words alone. Despite the care with which Fagan's phrases were formed, this was less than the improvement (2.2% and 22.7%) provided by very simple statistically defined phrases. Furthermore, Sembok's system [Sem89] achieved similar results to Fagan using only a very simple noun phrase grammar. Smeaton's method [Sv88] provided a somewhat smaller improvement over single word indexing than the above two systems, but required parsing only of noun phrases in requests, followed by looking for co-occurrence of phrase components in documents.

In summary, experiments on syntactic phrase formation have not found it superior to simple statistical techniques for phrase formation, and have not found much correlation between the sophistication of phrase formation and the resulting effectiveness improvements. This has led some researchers to suggest that current NLP technology is simply not powerful enough to be of use in indexing for IR [SS89]. We believe that such a conclusion is premature without a better understanding of why attempts to index on syntactic structures have failed to yield significant improvements in effectiveness.

4.2 Why Syntactic Phrase Indexing Hasn't Worked

Syntactic phrase indexing techniques produce an alternative representation of text, one which in most experiments has been combined with a word-based representation, in the hopes of providing effectiveness superior to either representation

1. Using a syntactic parser and grammar, produce a parse tree for a sentence.
2. Select from the parse tree all pairs of words that satisfy specified structural configurations. Usually these configurations are chosen so that one word is the head of a noun phrase and the other a modifier of that head, though certain verbal constructions are processed as well.
3. Group together into a single indexing phrase all head-modifier pairs based on morphologically related words. For instance, Fagan's system would form the syntactic phrase *quer analys* (after stemming) from all the following sentences ([Fag87], p. 158):
   - *They designed a query analysis system.*
   - *They designed a system for analyzing the queries.*
   - *The system analyzing the queries is automatic.*
   - *The queries analyzed by the system are well-constructed.*
   - *They designed a system to analyze queries automatically.*
4. Indexing documents on those phrases whose collection frequency is not too high. (Restricting very frequent phrases was found to have a small beneficial effect on effectiveness.)

Figure 4.1 Fagan's strategy for syntactic phrase indexing.

alone. In all cases, retrieval effectiveness with this new representation has been found to be the same or only slightly better than with a word-based representation.

To gain insight into this, recall some of the characteristics that we concluded in the previous chapter were desirable in a text representation:

1. *Small number of indexing terms.*
2. *Flat distribution of values for an indexing term.*
3. *Lack of redundancy among terms.*
4. *Low noise in indexing term values.*
5. *Lack of ambiguity for linguistically derived terms.*
6. *Terms should be related to the classes to be induced.*

If we consider how syntactic indexing phrases are formed, it becomes apparent that they will tend to fare badly on the first four of these criteria. If there are d

distinct words in the collection, then there potentially would be d^2 attributes in the indexing phrase representation. We have already commented on how even d tends to be much too large to allow effective learning from examples in IR systems, and the problem is even worse with d^2 attributes.

Almost all of these d^2 attributes will take on the value 0 for all documents, and even more will take on the value 0 for almost all documents. This is a consequence of the fact that the total number of attribute-value pairs with non-zero value will be about the same for this representation as for a free-text representation, even though the number of attributes has increased immensely. The root cause of this, in turn, is the fact that the number of pairs of grammatically related words in a sentence is linear in the number of words in the sentence.[1] There are other ways of using syntactic information that would change the particulars of this calculation, but not the basic problem.

Obviously, there are relationships among words that assure that certain pairs are much more likely than others. But in general very few pairs will have a frequency high enough to give them a good term discrimination value, to use Salton's measure. This is the conclusion at which Fagan ([Fag87], pp. 185–189) arrived: that syntactic phrases simply had too low frequency to be good identifiers.

Syntactic indexing phrase representations are also redundant and noisy. The causes of this are synonymy (several words can represent the same meaning) and compositionality (meanings of natural language structures are often formed directly from the meanings of their component parts). If each of the words in a particular phrase has k synonyms, then it is reasonable to expect that there will be at least k^2 phrases that could have the same meaning. A phrase representation is therefore highly redundant.

This redundancy is hidden by what can be viewed as a high noise level. If two phrases do have identical meanings then ideally we would want them to be assigned to all the same documents. However, the opposite is likely to be the case with natural language text. In short texts such as abstracts, there simply will not be occasion to refer to a concept enough times to use all the synonymous phrases for it. In longer documents the tendency of the author and communities of authors to reuse certain phrases will interfere with all synonymous phrases being assigned. One way to view this problem is as a kind of noise that converts non-zero values for terms into zero values, especially when synonymous terms are present.

On the characteristic of ambiguity, syntactic indexing phrases stack up much better. Each of the words in a phrase provides a context that limits the meanings of the other words in the phrase. Syntactic context has been used for disambiguation in computational and corpus linguistics [Atk87, Hir87] as well as in IR [Ear73, VBRV87].[2]

---

[1] Sentence structures involving conjunction can produce more than a linear number of pairs of grammatically related words, but this does not significantly increase the total number of pairs.

[2] A wide variety of evidence besides syntactic context can be used in disambiguation, as discussed by Krovetz [KC89].

To give an example, consider the word *group*. In a collection of abstracts from, say, *Communications of the ACM*, this word can mean a collection of entities, a collection of people in particular, a mathematical construct, or the action of collecting together things. However, if we look at some of the words that *group* has direct syntactic connections with, we see that it is less ambiguous in context:

- COLLECTION OF OBJECTS: *group of items, large group, another group, group of people, group of lines, linkage into groupings, group of subjects*
- COLLECTION OF PEOPLE: *systems group, work of group, scheduling group* (i.e. we were scheduling the group), *group has written, ALCOR group, goals for group, group in project*
- TO COLLECT: *group records* (to group records), *group variables* (to group variables), *grouping requests* (i.e. we were grouping requests)
- MATH STRUCTURE: *automorphism groups, finite groups, group theory, symmetric group*

To the extent meanings of syntactic structures are compositionally formed from the meanings of component words, unambiguous words imply unambiguous phrases.

Besides being less ambiguous than individual words, syntactic indexing phrases for the most part have meanings that are narrower than those of their component words. If we take the view that the meaning of a word is a set of entities in the universe, then the set of entities corresponding to a phrase is almost always smaller than the set corresponding to either of the individual words. For example *silicon chip* is more specific than *chip* or *silicon thing*, and *Jones proved* is more specific than *anyone proved* or *Jones did something*.[3] We might therefore hope for a better match between the underlying concepts for phrases and those for user requests to text retrieval systems, which tend to be much narrower in scope than the concepts corresponding to individual words. The fit for text categorization and text filtering systems would depend on the category or profile set.

Given the extent of the problems with syntactic indexing phrase representations, it is not surprising that they have not resulted in significant effectiveness improvements. However, it is crucial to note that the problems with phrases are for the most part statistical, and that these indexing terms have good semantic properties. Fortunately, more is understood about improving the statistical properties of representations than about improving their semantic properties. In the next section we propose a scheme for improving the statistical properties of a syntactic indexing phrase representation while keeping its semantic properties intact.

---

[3]This is an extremely simplified view of the complexities of natural language semantics.

4.3 Dimensionality Reduction

In the previous section we pointed out that, considered as a feature set, syntactic indexing phrases suffer from high dimensionality, skewed feature value distributions, redundancy, and noise. These characteristics suggested that we investigate the use of a dimensionality reduction technique. The major dimensionality reduction techniques that have been explored in IR have been:

1. feature selection (term selection)
2. factor analysis (latent indexing)
3. pattern clustering (document clustering)
4. feature clustering (term clustering)

Of these, we rejected factor analysis on the basis of computational expense, given the very large sets of features produced by syntactic phrase indexing. Document clustering would affect most or all of the problem characteristics of an indexing phrase representation, probably beneficially. However, it is difficult to predict or control these changes, because the effect on the text representation is indirect, via changes in what is considered a unit of text to be indexed. In addition, documents cannot be clustered without some existing text representation, which raises questions about the interaction of that representation and the phrasal one we want to improve.

Feature selection and term clustering are the techniques that most directly alter the properties of a text representation. Feature selection would have reduced the dimensionality and, potentially, the redundancy of a syntactic indexing phrase representation, by selecting only a subset of the phrases to use. This would not, however, change the skewed feature value distribution, since each phrase would retain its set of assignments. Nor would it have improved the problem of inconsistency of feature assignment.

Term clustering, on the other hand, had the potential to address all four problems. The point of term clustering is to recognize redundancies between terms and cluster these terms. By replacing individual terms with clustered terms, dimensionality would be reduced. By assigning the clustered term to all documents to which the component terms were assigned, noise would be reduced. Finally, the clustered term would take on non-default values in more documents that the individual term, so the skew in feature value distribution would be reduced.

Another reason we suspected term clustering might help is the results by Fagan [Fag87] and Salton ([SM83], p. 286) showing proximity phrases outperforming syntactic phrases. It seemed unlikely that proximity phrases were superior on the basis of having a more coherent meaning. Two words that merely occur within some specified number of words of each other could have a wide variety of syntactic and semantic relationships between their occurrences. We believed it was more likely that proximity phrases are superior on the basis of having a higher frequency than syntactic phrases, even if this higher frequency comes at the cost of including

some inappropriate syntactic structures. If clusters were restricted to structures with related meanings then effectiveness should be even better. In addition, more predictable improvements might be achieved, in contrast with improvements that vary widely between test collections, as with proximity phrases [Fag87].

As we will report in subsequent chapters, the particular term clustering techniques we investigated in this dissertation did not yield significant effectiveness improvements in text retrieval and text categorization tests. We discuss in Section 10.6 why these predictions were not borne out. It is important to note that these negative results apply only to the traditional statistical term clustering techniques we experimented with.

In the next section, we review previous work on term clustering, showing that clustering of syntactic phrases was, before the work reported in this dissertation, a mostly unexplored approach to text representation.

4.4 Term Clustering

Term clustering is the application of feature clustering to text representations. The idea is to detect groups of terms within a text representation that have the same or related meanings. These groups can be used to replace or supplement the original terms in the text representation. Most research on term clustering has been done in the context of text retrieval systems, and much of it has been driven by the desire to improve the recall of text retrieval systems, by allowing requests to match documents with similar content but dissimilar words.

Any clustering algorithm requires some way to measure the similarity or dissimilarity of the items to be clustered. Typically this is done by associating a set of feature values with each item, and then measuring how similar the feature values are for pairs of items. In feature clustering, the items to be clustered are themselves features of other objects. For clarity we will use the term *metafeatures* to refer to the features of features. Metafeatures for term clustering will be discussed in the next section.

4.4.1 Term Clustering of Words

The largest body of research on term clustering was performed by Sparck Jones [Spa71, SB71, Spa73a]. She investigated the effect of varying clustering strategies, term similarity measures, and vocabulary characteristics on the text retrieval effectiveness achieved with a clustered representation. Some of her most important conclusions were that clusters should be restricted to relatively infrequent and highly similar terms, clusters should be used to supplement the original terms rather than replace them, and that clustering was unlikely to be effective if the relevant and non-relevant documents were not well separated on the input representation. The particular shape of clusters formed and the particular measure of similarity between terms was not found to have a significant effect. Of the several collections she experimented with, only one had its retrieval effectiveness significantly improved by term clustering.

Similar early experiments were performed by Salton and Lesk [SL68], Lesk [Les69], and Minker, et al [MWZ72]. Salton and Lesk compared statistical term clustering with manually constructed thesauri on three test collections. No significant effectiveness improvements were found for statistical term clustering, in comparison with significant improvements for two out of three collections for the manual thesauri.

Lesk's experiments were, strictly speaking, with association lists rather than clusters, the difference being that a term A can be considered similar to a term B without the reverse holding. Lesk expanded both queries and document descriptions with similar terms of moderate collection frequency, but achieved no large effectiveness improvements. Lesk studied the term similarities that were actually produced and concluded that the small size of his collections (40,000 to 110,000 words) meant that the similarities were local to the collections, and were not good indications of the general meanings of the words.

Minker and colleagues experimented with two collections, and with three different text representations for each. Terms from all six representations were clustered using a variety of graph-theoretic algorithms. Like Sparck Jones, Minker found that small clusters performed the best, but he found no significant effectiveness improvements over indexing on terms.

All of the above researchers used co-occurrence in documents as the basis for term similarity. This means that each metafeature corresponded to a document, and the metafeature value was 1 or 0 depending on whether the term occurred in document associated with the metafeature. Other sources for metafeatures in term clustering have been co-occurrence in syntactic relationships with particular words [HGS75] and presence in pairings between queries and relevant documents [YR77]. Crouch recently achieved significant effectiveness improvements on two collections by first clustering documents, and then grouping low frequency terms that occurred in all documents of a document cluster [Cro88].

4.4.2 Integration of Syntactic Phrase Indexing and Clustering

While there has been extensive research on both term clustering and syntactic phrase indexing, the two techniques have not been directly combined before. Of course, almost all phrase generation systems in effect do some clustering when they normalize phrases, mainly through stemming. The FASIT system [DG83] combined all phrases that had a particular word in common into a group, a simple form of clustering that did not appear to be effective. Antoniadis, et al describe a similar method, but it is not clear if it was actually used in their system [ALBPR88].

More traditional statistical clustering techniques have been used in at least two IR interfaces to suggest terms, including syntactically formed phrasal terms, that a user might want to include in their request. The LEADER system [HK69] formed cliques of phrases based on co-occurrence in full document texts, and the REALIST system used unspecified statistical techniques to provide lists of strongly correlated terms [Thu86]. Neither study presented any effectiveness data resulting from the use of these strategies, however.

Salton [Sal68] investigated indexing documents on *criterion trees.* These were equivalent to hand-constructed clusters of syntactic structures, with individual words replaced by class labels from a manually constructed thesaurus. A related strategy is Sparck Jones and Tait's [ST84] generation of groups of alternative indexing phrases from a semantic interpretation of a request text. The phrases generated by this method only contained words from the request, but a thesaurus could have been used, as with criterion trees. Neither of these methods were tested on large enough collections to draw firm conclusions about their efficacy.

Lochbaum and Streeter have recently reported on the use of a factor analysis technique, singular value decomposition (SVD) for text retrieval [LS89]. SVD was used to detect and extract redundancies in the relationship between indexing terms and documents. It was found that the inclusion of some noun phrases in addition to single words improved the effectiveness achieved with the compressed representation. Since SVD operates by detecting and collapsing redundancies among terms (as well as among documents), this result is suggestive that term clustering of phrases will provide an improved representation.

4.4.3 Discussion

Applying term clustering to improve syntactic phrase indexing made sense only if term clustering itself was understood well enough that it could be applied plausibly to a new class of indexing terms. Previous research in the area of term clustering revealed a few important guidelines, and considerable evidence that many other choices are not of much significance. With respect to term clustering, the particular clustering algorithm used has not been found to make much difference, as long as clusters are small and composed of low frequency terms. This is fortunate, since any one set of experiments could only explore one or a few clustering algorithms.

On the other hand, there was some evidence that the metafeatures taken into account in computing the similarity between terms can have an important effect. Crouch's strategy, for instance, partially addressed the dilemma that infrequent terms are the very terms that most benefit from clustering, but are also the most difficult on which to get accurate co-occurrence data. Again, any one set of experiments would only be able to explore a few metafeature definitions, and here there was reason to believe that results would not be applicable to other metafeature definitions. However, Crouch's results gave some encouragement that at least some metafeature definitions were reasonable.

So, despite a lack of significant effectiveness improvements in previous research, it appeared that a good deal was known about better and worse approaches to statistical term clustering. It therefore appeared to be reasonable to investigate this strategy for improving the effectiveness of a syntactic indexing phrase representation.

It is important to note that only a limited set of approaches to forming terms corresponding to groups of syntactic phrases were tested, and to a considerable extent found ineffective, in the research reported in the remaining chapters. Many other approaches to forming groups of syntactic phrases would also have beneficial effects on the statistical properties of the syntactic phrase representation, and have not yet been tested. Some of these approaches will be mentioned in Chapter 11.

4.5 Conclusion

By examining syntactic phrase indexing from a machine learning standpoint, we found that the disappointing results of previous experiments are not surprising. However, the characteristics of syntactic phrases that are most problematic for text classification—high dimensionality, skewed feature value distribution, redundancy, and inconsistency of assignment—appeared susceptible to improvement by statistical dimensionality reduction methods. In surveying the literature on applications of one such dimensionality reduction technique, term clustering, we found that it too had not led to significant effectiveness improvements in the past. However, it had not been applied to a syntactic phrase representation before, and it appeared that the two techniques are complementary.

Therefore, the following hypothesis guided the research reported in the subsequent chapters of this dissertation:

> Main Hypothesis: A text representation consisting of clusters of syntactic phrases will allow more effective text classification than a representation consisting of individual words or individual syntactic phrases.

We turn in the remaining chapters to our experiments investigating whether using both techniques actually leads to significant improvements in text clasification effectiveness. Given that previous research on both techniques focused almost exclusively on text retrieval, as opposed to text categorization or document clustering, our first experiments investigated combining syntactic phrase indexing and statistical term clustering for text retrieval.

CHAPTER 5

REPRESENTATION QUALITY IN TEXT RETRIEVAL: EXPERIMENTS

The research reported in this chapter was our first investigation of the hypothesis that term clustering would improve the effectiveness of a syntactic indexing phrase representation for text classification systems, and in particular for text retrieval systems. We begin by discussing the specifics of our syntactic phrase generator and the phrases formed. We then turn to the clustering of phrases. While the low frequency of occurrence of phrases makes them desirable to cluster, it also introduces problems for traditional similarity measures based on co-occurrence in documents. We instead formed clusters based on co-occurrence in groups of documents defined by controlled vocabulary indexing.

We present a variety of experiments varying specifics of the clustering process in ways found in the past to be important to the effectiveness of term clustering. For the most part results from experiments on clustering words were found to hold for clustering phrases as well.

These initial experiments produced only small effectiveness improvements, and suggested that larger corpora would be necessary, if not sufficient, to produce high quality phrase clusters by statistical term clustering.

5.1 Extracting Syntactic Phrases

This section first describes a particular goal for phrase formation and how our system approximated this ideal. We then show some of the strengths and weaknesses of the system by the analysis of an example sentence. Finally, we present statistics on phrase formation for the CACM-3204 corpus.

5.1.1 Syntactic Analysis Technology

One factor that makes previous research on syntactic indexing hard to evaluate is the wide range of heuristic techniques used in generating syntactic indexing phrases. Since none of these variations has proven strikingly superior to others, we opted for a definition of phrases that was as simple as possible linguistically. We defined a syntactic indexing phrase to be any pair of non-function words in a sentence that were heads of syntactic structures connected by a grammatical relation. Examples are a verb and the head noun of a noun phrase that is its subject, a noun and

a modifying adjective, or a noun and the head noun of a modifying prepositional phrase. This is essentially the definition used by Fagan [Fag87], except that we form phrases from all verbal, adverbial, and adjectival constructions, and do not maintain exclusion lists of specially treated words.

It is important to distinguish the definition of syntactic phrases used by a system from the actual set of phrases produced. Current syntactic analysis systems are far from perfect, so any definition of syntactic phrases that is not of the form "syntactic phrases are what my program produces" can only be approximated. Even the PLNLP parser used by Fagan produced a correct analysis of only 32% of a typical set of sentences [SS89], and that system was the result of a large-scale corporate development effort.

In designing our phrase generation system we attempted to generate all phrases that suited our definition, while avoiding the complexity and ambiguity of producing a full parse for each sentence. Our approach was to parse only the constituents of a sentence below the clause level. The analysis of a sentence, therefore, was a sequence of noun phrases, adjective phrases, adverb phrases, verb groups, and miscellaneous punctuation and function words. Since much of the complexity in typical natural language grammars is in rules to capture clause level structure, we were able to restrict ourselves to a grammar of only 66 rules.

Limiting the complexity of analysis does not limit the need for a large lexicon, since every word still had to be interpreted. We used the machine-readable version of the Longman Dictionary of Contemporary English (LDOCE) [BB87], which provided syntactic categories for about 35,000 words. A morphological analyzer for inflectional suffixes was used to extend the effective vocabulary of the system. Even so, a substantial number of words encountered in text were not present in the dictionary. These tended to be compound words, proper nouns, or very technical terms. These unknown words were assumed to be ambiguous between the categories noun, verb, and adverb, and were allowed to be disambiguated during parsing.

Parsing was performed by a chart parser operating in bottom-up mode.[1] The bottom-up parsing strategy produced a large number of overlapping parse trees covering parts of the sentence. The parser then selected a small set of non-overlapping trees that together covered the entire sentence. Phrase formation used these trees in two ways. Phrases were generated from complete constituents by means of annotations to each grammar rule. These annotations indicated which components of a tree corresponding to that rule should be combined into a syntactic indexing phrase.

It sometimes was desirable to produce phrases from neighboring constituents as well. For instance, if a verb group was followed by a noun phrase, we wanted to combine the verb with the head noun of the noun phrase, on the assumption that such a noun phrase was often the syntactic object of the verb. Heuristics for forming phrases under these circumstances, including the handling of conjunction,

[1]The parser was designed and implemented at the University of Massachusetts by John Brolio, who also was the principal designer of the syntactic grammar.

punctuation, and function words, were encoded in a small (5 state) pushdown automaton.

Note that the two words in a phrase were considered to be unordered, and no distinction was made between phrases formed from different syntactic structures.

5.1.2 An Example of Phrase Generation

As an example, consider the following sentence from the CACM-3204 collection:

> *Analytical, simulation, and statistical performance evaluation tools are employed to investigate the feasibility of a dynamic response time monitor that is capable of providing comparative response time information for users wishing to process various computing applications at some network computing node.*

A complete and correct analysis of this sentence would be extremely complex and would have to be distinguished from a large number of plausible alternatives. However, the partial syntactic constituents produced by our system capture much of the structure necessary to produce reasonable phrases. The greatest advantage of this approach is that reasonable analyses can be produced for any sentence. In Figure 5.1.2 we show the phrases that would be produced from a perfect parse of the sentence, and those that were produced by our system. Bracketed phrases are ones that would not have been produced by a perfect system, though some are reasonable indexing phrases.

The phrases generated from this example sentence exhibit some of the strengths and weaknesses of our system. For instance, the words *analytical, statistical, evaluation*, and *feasibility* were not present in the lexicon. Grammatical constraints were able to disambiguate *evaluation* and *feasibility* correctly to nouns, while *analytical* and *statistical* were incorrectly disambiguated to nouns. However, the incorrect disambiguations did not affect the generation of phrases, since premodifying nouns and adjectives are treated identically.

The presence of a word in LDOCE did not guarantee that the correct syntactic class would be assigned to it. The words *tool, dynamic, time, monitor, provide/providing, comparative, wish*, and *process* all had multiple syntactic classes in LDOCE. Of these, *dynamic, providing, comparative, wishing*, and *process* were disambiguated incorrectly. The only case where phrase generation was seriously impaired was in the interpretation of *providing* as a conjunction.[2] This meant that the phrases *providing information* and *providing users* were not generated. The interpretation of *wishing* and *process* as nouns, and the resulting interpretation of a clausal structure as a noun phrase, while atrocious from a linguistic point of view, had a relatively minor effect on phrase generation.

---

[2]One price of using a machine-readable dictionary as a syntactic lexicon is the occasional odd classification.

DESIRED PHRASES	PHRASES PRODUCED
analytical tools	*<analytical employed>*
simulation tools	*<employed simulation>*
statistical tools	*statistical tools*
performance evaluation	*<performance tools>*
evaluation tools	*evaluation tools*
tools employed	*tools employed*
employed investigate	*employed investigate*
investigate feasibility	*investigate feasibility*
feasibility monitor	*feasibility monitor*
response time	*<response monitor>*
time monitor	*time monitor*
dynamic time	*<dynamic monitor>*
monitor capable	
capable providing	*<capable feasibility>*
providing information	*<capable information>*
comparative information	*comparative information*
response time	*<response information>*
time information	*time information*
information users	*<information wishing>*
	<information applications>
users wishing	*users wishing*
wishing process	*<wishing applications>*
process applications	*process applications*
various applications	*various applications*
computing applications	*computing applications*
process node	*<applications node>*
	<wishing node>
network node	*network node*
computing node	*computing node*

Figure 5.1 Desired and actual phrases for example sentence. Bracketed phrases should not have been produced.

Table 5.1 Statistics on phrase generation for 1,425 CACM documents.

Collection Frequency (in 1,425 Docs)	Unstemmed		Stemmed	
	No. Distinct Phrases	Total Phrase Occurrences	No. Distinct Phrases	Total Phrase Occurrences
1	41,500	43,612	32,470	34,689
2	3,399	7,336	4,056	8,866
3	906	3,015	1,284	4,299
4	370	1,687	576	2,584
5	169	963	309	1,735
6	124	850	218	1,503
7	57	443	108	855
8	47	458	90	814
9+	128	2,157	281	5,176
Total	46,700	60,521	39,392	60,521

5.1.3 Phrase Statistics

For the experiments reported here we parsed and generated phrases from the titles and abstracts of 1,425 documents, totaling 110,198 words, from the CACM-3204 collection. We used only those documents that have *Computing Reviews* categories assigned to them, since our clustering strategy required that controlled vocabulary indexing be available for documents. Table 5.1 breaks down the phrases generated according to the number of times they occurred in these 1,425 documents.

As expected, the number of phrases was very large, and few phrases had many occurrences. We used the Porter stemmer [Por80] to stem the words in phrases, which increased phrase frequency somewhat. These stemmed phrases were used for all the experiments reported here.

5.2 Clustering Phrases

Given the few differences found between text representations produced by different clustering algorithms, we chose to form the very simple clusters that Sparck Jones referred to as *stars* [SB71]. These clusters consist of a seed item and those items most similar to it. A fixed number of nearest neighbors, a minimum similarity threshold, or both can be used. Here are some randomly chosen example clusters formed from CACM phrases when clusters were restricted to a size of 4:

{ *linear function, comput measur, produc result, log bound* }

{ *princip featur, draw design, draw display, basi spline, system repres* }

{ *error rule, explain techniqu, program involv, key data* }

{ *substant increas, time respect, increase program, respect program*}

The seed phrases are underlined above. Some clusters contain more than 4 elements, since elements with negligibly greater dissimilarity to the seed than the fourth element were also retained.

The clusters formed rarely contained any exact synonyms for the seed phrase. This is not surprising since, of the large number of phrases with a given meaning, one will usually be considerably more frequent than the others. Given the relatively small size of the CACM corpus, only the most frequent of the synonymous phrases will have more than one occurrence. Since we required that a phrase must occur at least in at least two documents to be clustered, synonymous phrases were almost never clustered. However, some good clusters of closely related phrases were formed, along with many accidental clusters of essentially unrelated phrases.

The rest of this section discusses how clusters were formed and how they were used in scoring documents. Section 5.3 will then discuss our experimental results.

5.2.1 Co-occurrence In Controlled Vocabulary Indexing Categories

The dilemma between the desire to cluster infrequent terms and the lack of information on which to judge their similarity is even more severe for phrases than for words. Given that only 1.8% of the distinct phrases in our corpus occurred more than 5 times, it was unreasonable to expect that many phrases would have any substantial number of co-occurrences in documents.

Crouch's strategy of looking for co-occurrence in document clusters was a promising alternative, but we knew that document clustering does not necessarily produce meaningful clusters. Therefore, instead of producing document clusters, we made use of the document clustering implicit in the controlled vocabulary indexing of the CACM collection. A total of 1,425 of the CACM documents are indexed with respect to a set of 201 *Computing Reviews* (*CR*) categories [FNL88, Lew90]. Of those categories, 193 are assigned to one or more documents. Since *CR* categories are arranged in a three-level hierarchy, we assumed that whenever a document was assigned to a category it was also assigned to all ancestors of that category.

This means we had available a set of 193 metafeatures for phrases, each corresponding to a *CR* category. We then needed a method for deciding what the value of each metafeature was for each phrase. Crouch found the set of low frequency terms in each of the documents in a cluster and took the intersection of these sets. This corresponds to using binary valued metafeatures and a very strict clustering method.

The large and quite variable size of the *CR* clusters makes a binary metafeature approach questionable, since presence in one document out of two assigned to a *CR* category seems more important than presence in one document out of one hundred assigned to a category. Metafeatures that took on a value between 0 and 1 seemed more appropriate. We used a metafeature value of n_{pc}/n_c, where n_{pc} was the number of occurrences of phrase p in category c, and n_c was the total number of occurrences of all phrases in category c. This treated multiple occurrences of a phrase as being more significant than single occurrences, and also normalized for the large differences in the number of documents, and thus phrases, appearing in the different categories.

The cosine correlation was used to compute the similarity between metafeature vectors for different phrases. This had the effect of normalizing for overall phrase frequency. All phrases occurring in 2 or more documents were used in clustering, expect when otherwise mentioned in results.

5.2.2 Weighting of Clusters

The point of forming clusters, of course, was to use them in retrieval. This required a method for incrementing the scores of documents based on the presence of phrases and clusters of phrases in queries and documents. We chose to use the same weighting methods used by Fagan for phrases and by Crouch for clusters, since these methods had shown some effectiveness in the past.

Fagan [Fag87, Fag89] assigned a two-word phrase a weight (in both queries and documents) equal to the mean of the weights of its component stems. The stem weights themselves are computed as usual for the vector space model. The inner products were computed separately for stems and phrases and then added together, potentially with different weightings.

Crouch [Cro88] used a very similar method for clusters, giving them a weight in a query (or a document) equal to the mean of the weights of the cluster members in the query (or the document). The resulting weights were then multiplied by 0.5 in both documents and queries, for an overall downweighting factor of 0.25 for clusters with respect to terms.

Combining these gave the following similarity function to be used for ranking documents:

$$SIM(q,d) = (c_s \cdot ip(q_s, d_s)) + (c_p \cdot ip(q_p, d_p)) + (c_c \cdot ip(q_c, d_c))$$

where ip is the inner product function, q_s, q_p, and q_c are the weight vectors of stems, phrases, and phrase clusters for queries, d_s, d_p, and d_c are the vectors for documents, and c_s, c_p, and c_c are the relative weights of stems, phrases, and phrase clusters.

5.3 Experiments

The main goal of the experiments reported here was to discover whether applying statistical term clustering to syntactic indexing phrases from a small corpus would result in an improved text representation for text retrieval. Another goal was to explore whether the factors that have been found to be most important in clustering of words would have the same impact on clustering of phrases. These include the size of clusters formed, the frequency of items clustered, and the maximum dissimilarity tolerated between cluster members.

All retrieval results are based on the full CACM collection of 3,204 documents. We used only the 50 requests that do not ask for documents by particular authors, and for which there are one or more relevant documents.

Table 5.2 Effectiveness using phrase clusters and stems.

Recall Level	Precision					
	Clusters + Stems				Phrases	
	Size 2	Size 4	Size 8	Size 12	+ Stems	Stems
0.10	55.5	55.5	57.9	57.1	58.1	56.3
0.20	43.2	42.0	42.2	41.9	45.4	41.0
0.30	37.7	37.0	36.5	36.2	38.0	35.7
0.40	31.1	30.5	30.8	30.0	30.2	29.6
0.50	23.3	23.3	22.2	22.3	23.4	22.0
0.60	19.5	19.3	18.2	18.3	19.0	18.8
0.70	13.5	13.3	13.3	13.3	13.7	13.8
0.80	9.2	9.4	9.4	9.3	9.5	9.9
0.90	5.5	5.8	5.6	5.6	5.6	6.1
1.00	4.2	4.1	4.1	4.1	4.1	4.7
Avg. Prec.	24.3	24.0	24.0	23.8	24.7	23.8
Change	+2.1%	+0.8%	+0.8%	+0.0%	+3.8%	

5.3.1 Effectiveness of Syntactic Phrase Clusters

Our first concern was whether the clusters of syntactic phrases formed from this small corpus would be sufficient to improve retrieval effectiveness. Table 5.2 compares recall and precision figures for four sizes of clusters to the figures for word stems and for stems combined with syntactic phrases. Clusters produce a smaller improvement than phrases, and neither is significantly better than the use of stems alone.

Using both clusters and phrases (Table 5.3) provides the most improvement. These results would be classified as "noticeable" (> 5.0%) but not "material" (> 10.0%) according to Sparck Jones' criteria [SB77]. We investigated varying the weighting of the cluster and phrase vectors (c_c and c_p, respectively), but found only trivial and inconsistent improvements resulting from any values besides 1.0. In particular, reducing weighting of clusters to Crouch's value of 0.25 caused a small decrement in effectiveness, providing some evidence that clusters of phrases are better content indicators than clusters of words.

5.3.2 Factors Affecting Phrase Clustering

In our survey on term clustering, we mentioned a number of factors that had been found in the past to impact the effectiveness of term clustering. We have already mentioned the effect of cluster size. Sparck Jones found small, tight clusters, of size 2 to 4, to be most effective, and our results are in agreement with this. We also found that using clusters of phrases in addition to phrases, rather than instead of phrases, was most effective. This again is in agreement with Sparck Jones' results on clustering of stems.

Table 5.3 Effectiveness using phrase clusters, phrases, and stems.

Recall Level	Precision					
	Clusters + Phrases + Stems				Phrases + Stems	Stems
	Size 2	Size 4	Size 8	Size 12		
0.10	57.4	60.0	59.3	58.5	58.5	56.3
0.20	46.4	46.4	46.1	45.0	45.4	41.0
0.30	38.8	39.5	38.9	37.7	38.0	35.7
0.40	31.3	31.1	31.1	30.8	30.2	29.6
0.50	23.0	23.1	23.1	23.1	23.4	22.0
0.60	19.3	19.5	19.5	19.5	19.0	18.8
0.70	13.9	13.9	13.8	13.7	13.7	13.8
0.80	9.6	9.8	9.7	9.6	9.5	9.9
0.90	5.7	5.7	5.7	5.7	5.6	6.1
1.00	4.2	4.2	4.2	4.2	4.1	4.7
Avg. Prec.	25.0	25.3	25.1	24.8	24.7	23.8
Change	+5.0%	+6.3%	+5.5%	+4.2%	+3.8%	

Another approach to forming tight clusters would be to require that phrases have no greater than a fixed dissimilarity with the seed phrase. This causes some phrases not to cluster at all. We investigated several dissimilarity thresholds for cluster membership, but found only trivial improvements, and some degradations, in effectiveness.

Another factor that has been found to impact term clustering is the frequency of the terms being clustered. The exclusion of high frequency terms from clusters was found by Sparck Jones in particular to be important in achieving an effective term clustering. Maximum frequency thresholds used by Sparck Jones included 20 out of 200 (10%) documents, 20 out of 541 documents (3.6%), and 25 out 797 documents (3.1%) [Spa73a].

Since only 8 stemmed phrases occurred in more than 45 (3.2%) of the 1,425 documents used for clustering, it was questionable whether omitting frequent phrases would be useful. We experimented with forbidding phrases that occurred in more than 45 documents from participating in clusters, and found this actually produced a slight decrease in effectiveness. Forbidding phrases occurring in more than 30 documents produced a larger decrease. Examining the 8 phrases of frequency greater than 45 shows that even here there are several which are moderately good content indicators (*oper system, comput program, program languag, comput system, system design*) as well as several fairly bad ones (*paper describ, paper present*, and *present algorithm*). Therefore, omitting the most frequent phrases does not appear to be an appropriate strategy when clustering phrases.

One can also argue that very infrequent phrases should be omitted from clusters. If a term does not occur a sufficient number of times then we will have not have enough data on its distribution to cluster it accurately. Most work on term clustering

has required that terms occur in two or more documents to become part of a cluster, but higher thresholds conceivably could result in more accurate clusters.

We investigated requiring that phrases occur in at least 3, 4, 5, or 6 documents in order to be clustered. These were fairly severe restrictions considering the low frequency of phrases, resulting in reducing the number of phrases available for clustering from 6922 to 2866, 1582, 1015, and 706 respectively. Small effectiveness improvements resulted for some of these restrictions in combination with some cluster sizes. However, the improvements vanished when clusters were used in combination with phrases as well as stems. These results do help confirm that the small amount of frequency data available on phrases was a major impediment to forming effective clusters.

5.4 Analysis

In this section we report on a range of possible reasons for the failure of syntactic phrase clusters to improve retrieval effectiveness significantly. Our goal was to discover what the most significant influences were on the effectiveness of syntactic phrase clusters, and suggest what new approaches might be pursued.

5.4.1 Document Scoring Method

The first possibility to consider is that there was nothing wrong with the clusters themselves, but only with how we used them. In other words, the coefficients of the classification functions (queries) derived from requests, or the numeric values assigned to the cluster attributes, might have been inappropriate. There is some merit in this suggestion, since the cluster and phrase weighting methods currently used are heuristic, and are based on experiments on relatively few collections.

On the other hand, scoring is unlikely to be the only problem. Simply examining a random selection of clusters, such as those listed in Section 5.2, shows they leave much to be desired as content indicators. We therefore need to consider reasons why the clusters formed were inappropriate.

5.4.2 Statistical Problems

The simplest explanation for the low quality of clusters is that not enough text was used in forming them. Table 5.1 gives considerable evidence that this is the case. The majority of occurrences of phrases were of phrases that occurred only once, and only 17.6% of distinct phrases occurred two or more times. We restricted cluster formation to phrases that occurred at least twice, and most of these phrases occurred exactly twice. This means that we were trying to group phrases based on the similarity of distributions estimated from very little data. Church [CGHH89] and others have stressed the need for large amounts of data in studying statistical properties of words, and this is even more necessary when studying phrases, with their lower frequency of occurrence.

Another statistical issue arises in the calculation of similarities between phrases. We associated with each phrase a vector of metafeature values of the form $n_{pc}/\sum n_{qc}$,

where n_{pc} is the number of occurrences of phrase p in documents assigned to *Computing Reviews* category c, and the denominator is the total number of occurrences of all phrases in category c. This is the maximum likelihood estimator of the probability that a randomly selected phrase from documents in the category will be the given phrase. Problems with the maximum likelihood estimator for small samples are well known [FH89, CG89], so this may have had an adverse impact on cluster quality.

Another question is whether the clustering method used might have been inappropriate. Previous research in IR has not found large differences between different methods for clustering words, and all clustering methods are likely to be affected by the other problems described in this section, so experimenting with different clustering methods deserves lower priority than addressing the other problems discussed.

A final issue is raised by the fact that using clusters and phrases together produced effectiveness superior to using either clusters or phrases alone. One way of interpreting this is that the seed phrase of a cluster is a better piece of evidence for the presence of the cluster than are the other cluster members. This raises the possibility that explicit clusters should not be formed at all, but rather that every phrase be considered good evidence for its own presence, and somewhat less good evidence for the presence of phrases with similar distributions.[3] Again, investigating this is not likely to be profitable until other problems are addressed.

5.4.3 Weaknesses in Syntactic Phrase Formation

Another set of factors potentially affecting the effectiveness of phrase clustering is the phrases themselves. Our syntactic parsing is by no means perfect. Incorrect phrases could both cause bad matches between queries and documents, and interfere with the distributional estimates on which clustering is based.

It is difficult to gauge directly the latter effect, but we can measure whether syntactically malformed phrases seem to be significantly worse content identifiers than syntactically correct ones. To determine this we found all matches between queries and relevant documents on syntactic phrases. We examined the original request text to see whether the phrase was correctly formed or whether it was the result of a parsing error, and did the same for the phrase occurrence in the document. We then gathered the same data for about 20% of the matches (randomly selected) between queries and *non*relevant documents.

The results are shown in Table 5.4. We see that for both relevant and nonrelevant documents, the majority of matches are on syntactically correct phrases. The proportion of invalid matches is somewhat higher for nonrelevant documents, but the small difference suggests that syntactically malformed phrases are not a primary problem.

5.4.4 Correct Phrases with Poor Semantics

In proposing the clustering of syntactic phrases, we argued that the semantic properties of individual phrases were good, and only their statistical properties

[3]Ken Church suggested this idea to us.

Table 5.4 Syntactic correctness of query phrases and phrase occurrences in documents.

Query / Relevant Document Matches (229 Pairs Total)		
	Correct Phrase in Doc	Flawed Phrase in Doc
Correct Phrase in Query	84.3% (193)	6.6% (15)
Flawed Phrase in Query	3.5% (8)	4.8% (11)

Query / Nonrelevant Document Matches (424 Pairs in Random Sample)		
	Correct Phrase in Doc	Flawed Phrase in Doc
Correct Phrase in Query	77.6% (324)	13.0% (55)
Flawed Phrase in Query	4.5% (19)	5.0% (21)

needed improving. This clearly was not true, since phrases such as *paper gives* (from sentences such as *This paper gives results on...*) are poor indicators of a document's content.

We believed, however, that such phrases would tend to cluster together, and none of the phrases in these clusters would match query phrases. Unfortunately, almost the opposite happened. While we did not gather statistics, it appeared that these bad phrases, with their relatively flat distribution, proved to be similar to many other phrases and so were included in many otherwise coherent clusters. As mentioned earlier, we experimented with addressing this problem by omitting high frequency phrases, but this approach actually decreased effectiveness.

Fagan, who did the most comprehensive study [Fag87] of phrasal indexing to date, used a number of techniques to screen out low quality phrases. For instance, he only formed phrases that contained a head noun and one of its modifiers, while we formed phrases from all pairs of syntactically connected content words. Since many of our low quality phrases resulted from verb / argument combinations, we reconsidered this choice in the later experiments for this dissertation.

Fagan also maintained a number of lists of semantically general content words that were to be omitted from phrases or that triggered special purpose phrase formation rules. We chose not to replicate this technique, due to the modifications required to our phrase generator, and the lack of guidelines by Fagan on how to choose an appropriate exemption list for a particular corpus or domain.

We did, however, conduct a simpler experiment that suggests distinguishing between phrases of varying qualities will be important. A graduate student who

Table 5.5 Effectiveness with human-selected query phrases.

Recall Level	Precision					
	Clusters + Phrases + Stems				Phrases + Stems	Stems
	Size 2	Size 4	Size 8	Size 12		
0.10	60.7	61.9	61.5	61.4	61.4	56.3
0.20	45.8	45.9	45.9	45.9	45.2	41.0
0.30	40.6	40.3	39.8	39.8	39.5	35.7
0.40	34.2	33.4	33.5	33.5	33.2	29.6
0.50	25.0	25.1	25.2	25.2	25.3	22.0
0.60	19.8	20.7	20.7	20.6	20.9	18.8
0.70	13.8	14.6	14.5	14.6	14.6	13.8
0.80	9.4	10.2	10.0	10.0	10.0	9.9
0.90	5.6	6.3	6.2	6.3	6.2	6.1
1.00	4.2	4.9	4.9	4.9	4.9	4.7
Avg. Prec.	25.9	26.3	26.2	26.2	26.1	23.8
Change	+8.8%	+10.5%	+10.1%	+10.1%	+9.7%	

was not involved in the phrase clustering experiments identified for each CACM request a set of pairs of words he felt to be good content identifiers. We then treated these pairs of words just as if they had been the set of syntactic phrases produced from the request. This gave the results shown in Table 5.5. As can be seen, retrieval effectiveness was considerably improved, even though the phrases assigned to documents and to clusters did not change. (More results on eliciting good identifiers from users are discussed in work by Croft and Das [CD90].)

Given this evidence that not all syntactic phrases were equally desirable identifiers, we tried one more experiment. We have mentioned that many poor phrases had relatively flat distributions across the *Computing Reviews* categories. Potentially this very flatness might be used to detect and screen out these low quality phrases. To test this idea, we ranked all phrases that occurred in 8 or more documents by the similarity of their *Computing Reviews* vectors to that of a hypothetical phrase with even distribution across all categories.

The top-ranked phrases, i.e. those with the flattest distributions, are found in Table 5.6. Unfortunately, while some of these phrases are bad identifiers, others are reasonably good. More apparent is a strong correlation between flatness of distribution and occurrence in a large number of documents. This suggests that once again we are being tripped up by small sample estimation problems, this time manifesting itself as disproportionately skewed distributions of low frequency phrases. The use of better estimators may help this technique, but once again a larger corpus seems called for. Another approach might be to compare the distribution of a phrase in a corpus from a narrow domain with a corpus that is representative of the language as a whole. Phrases that have the same behavior in the corpus of interest as in a very general corpus are probably not good content indicators.

Table 5.6 Syntactic phrases with least skewed distribution across *Computing Reviews* categories, and number of documents they appear in.

Phrase	Similarity to (1...1)	No. Docs
describ paper	.61	57
algorithm present	.56	64
design system	.55	54
comput system	.54	75
system use	.52	43
paper present	.51	47
languag program	.48	71
describ system	.47	26
requir time	.47	22
data structur	.47	38
process system	.46	21
inform system	.45	26
oper system	.44	59
program use	.44	27
model system	.44	26
execut time	.43	28
problem solut	.43	45
requir storag	.42	24
techniqu use	.42	40
gener system	.41	22

5.5 Summary

The fact that phrase clusters provided small improvements in effectiveness was encouraging, but the most clear conclusion from the above analysis was that syntactic phrase clustering needed to be tried on much larger corpora. While the presence of other problems, particularly the low semantic quality of some phrases, suggested that larger corpora might not be sufficient for statistical term clustering to produce high quality phrase clusters, such corpora definitely appeared necessary.

Our need for a larger corpus posed some problems for evaluation, since the CACM collection is one of the larger of the currently available text retrieval test collections. Since significantly larger text retrieval test collections were not available, we considered other approaches for further experimentation. One was to form clusters on a corpus different from the one on which the retrieval experiments are performed. If the content and style of the texts are similar enough, the clusters should still be usable.

However, there were other problems besides small corpus size that resulted from using a text retrieval test collection to evaluate text representation strategies. Current text retrieval collections have only a small number of requests, which means

that only a small proportion of the terms in a new text representation will be evaluated at all. Those terms that do show up are usually represented in only one or two queries. Fox ([Fox83], p. 78) mentions this shortage of requests as interfering with the ability to evaluate phrasal indexing strategies. Until text collections with thousands or tens of thousands of requests are available, this problem will persist.

Finally, there is a problem inherent in using textual requests to evaluate complex text representations. It is relatively straightforward to decide what set of words a request is specifying for use in a query, though even here there are alternate views of stoplists, stemming, and so on. It is much more difficult to decide what the set of, say, syntactic indexing phrases specified by a request is. The syntactic structure of requests is often different from that of sentences in documents. For instance, requests often include lists of words or phrases with no syntactic connections between them. Therefore, there is no particular reason to assume that the same syntactic analysis strategy used for documents should be used on requests. The question becomes even more complex for clustered representations.

Selecting a subset of a text representation to use in a particular query based on a natural language request is an important capacity for text retrieval systems to have, and should be the subject of continued investigation. However, this is a separate issue from that of creating text representations, and introduces a substantial confounding factor into studies of text representation.

Given the above difficulties, we considered whether a different text classification task might be appropriate for our studies of text representation. In the next chapter we discuss the text categorization task and its advantages for the study of text representation.

CHAPTER 6

TEXT CATEGORIZATION

Text retrieval is not the only text classification task. Given some of the difficulties discussed in the previous chapter, we examined whether some other text classification task might provide a better vehicle for studying text representation.

Text categorization, the assignment of documents to one or more of a set of preexisting categories, is an obvious candidate. However, we found only one precedent for text categorization as a vehicle for studying text representation [BB64]. Therefore, a number of questions needed to be answered:

1. Importance: Is it interesting, from a scientific, engineering, or application viewpoint, what the impact of text representation on text categorization is?

2. Advantages: Does text categorization have significant advantages for studying text representation? Are any other text classification tasks even more advantageous?

3. Standardization: Do text categorization strategies exist that are widely enough used and understood to serve as a vehicle for studying text representation?

4. Evaluation: Do we know how to measure the effectiveness of text categorization runs, and from these measurements can we draw conclusions about the relative merits of text representations?

We will claim in the rest of this chapter that the answer to all these questions is, to one degree or another, positive, and that text categorization is an excellent alternative to text retrieval for studying text representation.

6.1 Importance of Text Categorization

The impact of text representation on text categorization is of interest for two reasons. First, results from text categorization experiments are of interest for the light they shed on other text classification tasks, in particular text retrieval. Methods for text categorization, particularly statistical methods, are very similar to those used in text retrieval and text routing. In particular, the same machine learning algorithms used for relevance feedback in text retrieval can be used in statistical text categorization systems.

Secondly, text categorization itself has considerable practical applications. In Chapter 1 we cited examples of three classes of such applications: as an indexing

mechanism for text retrieval, as a component in text understanding systems, and as end in itself when categorization of documents is of interest.

The most important and most controversial application is in indexing documents for text retrieval. As mentioned in Chapter 1, there is considerable support for the idea that manual assignment of subject categories for documents does not provide a better representation than the extraction of words from free text. On the other hand, some experimental results suggest combining free text and controlled vocabuary representations can result in better effectiveness than using either alone [KMT+82, FK88, Tur90]. In any case, there has been a substantial financial investment in manual controlled vocabulary indexing by large organizations, such as the National Library of Medicine. Such organizations are unlikely to discontinue their use of controlled vocabulary indexing, so the development of automated techniques to aid or replace this currently human-intensive process would be of considerable interest.

6.2 Advantages of Text Categorization

In discussing our results on the CACM collection we mentioned a number of problems with comparing text representations on a standard text retrieval test collection. Briefly, the problems were the relatively small number of features tested in existing collections of test requests, the uncertainty of the mapping between textual requests and features used in queries, and the small amount of training data these collections make available for feature extraction.

Text categorization as a vehicle for studying text representation addresses all three of these problems. The first two problems are addressed by the fact that text categorization can be successfully performed using automated feature selection. Such human intervention may improve effectiveness, but it is not neccessary for reasonable effectiveness, as shown by studies where feature selection was accomplished wholly or in part by automatic means [Fie75, HZ80, BFL+88, Lew91a]. The ability to factor human feature selection out of categorization means that the inherent properties of the feature set can be studied more clearly. Categorization effectiveness on each of several complete feature sets may be easily compared.

The problem of the amount of data available for feature formation is also addressed by text categorization. As discussed in the previous section, large amounts of human effort have been devoted to assigning controlled vocabulary categories to documents. The resulting large document sets provide a substantial opportunity for unsupervised learning for feature extraction.

Are there any other text classification tasks that might provide an even better vehicle for studying text representation? Text routing shares the same disadvantages as text retrieval, with the additional problem that standard test collections are currently lacking. Document clustering has some of the same advantages as text categorization, but it is difficult to decide whether the clustering formed with one text representation is better than another.

Of text classification tasks, text categorization seems best suited to studying text representation. Text categorization of course has its own disadvantages for studying text representation. What these are and how they may be addressed is discussed in the next two sections.

6.3 Standardization in Text Categorization

A strong argument in favor of studying text representation via text retrieval, as we did with the CACM collection, is that there are widely investigated and well understood techniques—probabilistic and vector space retrieval—for accomplishing this task. This means that text representations can be compared with some confidence that the retrieval method making use of them is not the source of unexpected variations in effectiveness or is so inadequate that effectiveness differences are uninteresting.

Do similar methods exist for text categorization, and if so do these techniques have the other advantages we sought, in particular the ability to factor out human intervention? While a range of techniques have been investigated for text categorization, it is certainly the case that this investigation has been less extensive than for text retrieval. There are not one or two widely used methods, as in text retrieval. This results in part from the fact that text categorization has been investigated by researchers from several fields, for a variety of real world applications with application-specific demands.

On the other hand, two classes of techniques have been investigated enough that we can have reasonable confidence in the broad details of their operation. First, a number of systems [VS87, Har88a, HANS90] have embodied techniques similar to those used in expert systems for classification or diagnosis [Cla85]. Knowledge engineers define one or more layers of intermediate conclusions between the input evidence (words and other textual features) and the output categories and write rules for mapping from one layer to another. While a promising approach to text categorization, the large human influence in such an approach makes it inappropriate for our purpose of comparing text representations.

Of more interest to us, techniques based on statistical correlations between occurrences of words and assignments of manual indexing categories on a large corpus have been widely investigated and have achieved good results. The particular statistical technique used has varied widely, and has included Bayesian classification [Mar61, KW75], summing conditional probabilities [Hoy73, HZ80], factor analysis [BB63, BB64, Bor64], fuzzy sets [COL83], linear regression [BFL+88], and ad hoc "adhesion" coefficients [Fie75, Har82].

Some of these techniques have been used alone, and others in combination with additional knowledge bases. Together, however, they suggest that the basic strategy of categorizing documents based on observing statistical associations between textual features and categories is reasonable.

Competing techniques have rarely been compared on the same corpus, so it is possible that some of the above techniques are significantly better than others. However, the differences in effectiveness so far reported in the literature are well within the range of variation that experience with text retrieval suggests could arise solely from the characteristics of different test collections. In addition, the one direct comparison of which we are aware, between Bayesian and factor analytic techniques [BB64], showed essentially identical effectiveness for the two techniques. So it seems likely that any of the above mentioned techniques would be reasonable for comparing

text representations. We can therefore feel confident in choosing techniques on other criteria, such as how well they are described in the literature or whether they are supported by a theoretical model.

One aspect that has not seen much exploration in previous work on statistical text categorization is the categorization of texts with respect to many categories at once. Some studies have dealt only with the assignment or nonassignment of a single category, or have assumed that each document will be assigned to exactly one category. Most of those that have dealt with assignment of multiple categories have treated each category as a completely independent decision.

This is not necessarily a problem—independence assumptions are often made in information retrieval research. What this means, however, is that little is known about how to choose parameter settings for statistical categorizers in order to achieve reasonable results on an entire set of categories at once.

The problem may be made clearer by looking at the analogous situation in text retrieval. Suppose that a text retrieval system assigns some score to each document. Can we choose a single threshold such that, if we retrieve for each query all documents above that threshold, we will get a given level of recall (or precision)? The answer, for most current statistical retrieval methods, is no. The range of scores for relevant documents vary wildly between queries. Even if we put a threshold on the rank of documents rather than on their scores, we are unlikely to achieve good overall effectiveness, given the wide variation in the number of relevant documents per query.

We discuss in Section 6.4.5 how this problem has been finessed in research on text retrieval systems. For text categorization systems, this problem needs to be addressed more directly, and any use of text categorization to compare text representations will also have to involve some experimentation with methods for multiple categorization.

6.4 Evaluation in Text Categorization

If we are to claim that one text representation is superior to another for text categorization, we need to be able to tell when the output of one text categorization run is more effective than another. Perhaps because text categorization has been investigated by a wide variety of researchers for a wide variety of purposes, effectiveness measures have tended to vary widely between studies, and some studies have been flawed by inappropriate evaluations.

On the other hand, most of the more careful evaluations of effectiveness in text categorization have used the same model of decision making by the categorization system, and this is same model underlying most effectiveness measures for text retrieval as well.

We begin by discussing this *contingency table* model, which motivates a small number of simple and widely used effectiveness measures. Complexities arise, however, in how to compute and interpret these measures in the context of a text categorization experiment. The bulk of the discussion concerns these complexities.

Table 6.1 Contingency table for a set of binary decisions.

	Yes is Correct	No is Correct	
Decides Yes	a	b	$a+b$
Decides No	c	d	$c+d$
	$a+c$	$b+d$	$a+b+c+d=n$

6.4.1 The Contingency Table

Consider a system that is required to make n binary decisions, each of which has exactly one correct answer (either Yes or No). The result of n such decisions can be summarized in a contingency table, as shown in Table 6.1. Each entry in the table specifies the number of decisions with the specified result. For instance, a is the number of times the system decided Yes, and Yes was the correct answer.

Given the contingency table, three important measures of the system's effectiveness are:

(1) recall $= a/(a+c)$

(2) precision $= a/(a+b)$

(3) fallout $= b/(b+d)$

Measures equivalent to recall and fallout made their first appearance in signal detection theory [Swe64], where they play a central role. Recall and precision are ubiquitous in evaluating text retrieval, where they measure the proportion of relevant documents retrieved and the proportion of retrieved documents that are relevant, respectively. For text retrieval, fallout measures the proportion of *non*relevant documents that are retrieved, and has also seen considerable use.

A decision maker can achieve very high recall by rarely deciding No, or very high precision (and low fallout) by rarely deciding Yes. For this reason at least the pair of measures recall and precision, or the pair of measures recall and fallout, are necessary to ensure a nontrivial evaluation of a decision maker's effectiveness under the above model.

Another measure sometimes used in categorization experiments is overlap:

(4) overlap $= a/(a+b+c)$

This measure is symmetric with respect to b and c, and so is sometimes used to measure how much two categorizations are alike without defining one or the other to be correct.

It is appropriate at this point to mention some of the limitations of the contingency table model. It does not take into account the possibility that different errors have different costs; doing so requires more general decision-theoretic models. The contingency table also requires all decisions to be binary. For some applications, it may be desirable for category assignments to be weighted rather than binary. This is likely to be the case when text categorization is used for automated subject indexing.

However, our interest in text categorization is less in a particular application than in seeing how its effectiveness changes under different text representations. Therefore, the contingency table model, which is uniform and abstracted from particular applications, seems appropriate.

6.4.2 Defining Decisions and Averaging Effectiveness

The contingency table model presented above is applicable to a wide range of decision making situations. In this section, we will first consider how text retrieval has been evaluated under this model, and then consider how text categorization can be evaluated under the same model.

In a text retrieval system, the basic decision is whether or not to retrieve a particular document for a particular query. For a set of q queries and d documents a total of $n = qd$ decisions are made. Given those qd decisions, two ways of computing effectiveness are available. *Microaveraging* considers all qd decisions as a single group and computes recall, precision, fallout, or overlap as defined above. *Macroaveraging* computes these effectiveness measures separately for the set of d documents associated with each query, and then computes the mean of the resulting q effectiveness values.

Macroaveraging has been favored in evaluating text retrieval, partly because it gives equal weight to each user. A microaveraged recall measurement, for instance, would be disproportionately affected by recall on requests from users who desired large numbers of documents.

An obvious analogy exists between categories in a text categorization system and requests in a text retrieval system. The most common view taken of categorization is that an assignment decision is made for each category/document pair. A categorization experiment will compare the categorization decisions made by a computer system with some standard of correctness, usually human category assignment. In contrast to evaluations of text retrieval, evaluations of categorization have more often used microaveraging than macroaveraging. Unfortunately, ad hoc variations have often been made in the averaging, perhaps because the authors were not aware of the underlying contingency table model.

Whether microaveraging or macroaveraging is more informative depends on the purpose of the categorization. For instance, if categorization is used to index documents for text retrieval, and each category appears in user queries at about the same

frequency it appears in documents, then microaveraging seems appropriate. On the other hand, if categorization were used to route documents to divisions of a company, with each division viewed as being equally important, then macroaveraging would be more informative. The choice will often not be clearcut. We assume microaveraging in the following discussion unless otherwise mentioned.

6.4.3 Arithmetic Anomalies

The above discussion assumed that computing the effectiveness measures is always straightforward. Zero denominators can arise in the effectiveness measures recall, precision, or fallout when there exist no correct category assignments, no incorrect category assignments, or when the system never assigns a category. All these situations are unlikely when microaveraging is used, but are quite possible under macroaveraging.

For evaluating text retrieval, Tague [Tag81] suggests either treating 0/0 as 1.0 or throwing out the query, but says neither solution is entirely satisfactory. For a categorization system, we also have the option of partitioning the categories and macroaveraging only over the categories for which these anomalies do not arise. Fuhr suggests using a very low non-zero value for precision of 0/0 [FK84]. An alternative to this would be macroaveraging separately under the assumption of $0/0 = 0.0$ and $0/0 = 1.0$, which would show how sensitive the results are to this anomaly. If there were a significant difference, microaveraging probably would be preferable.

6.4.4 Standard of Correctness

One of the advantages we have claimed for text categorization as an experimental vehicle is the availability of large amounts of categorized text to use in both training and evaluation of text categorization systems. Evaluation under the contingency table model requires that such correctly categorized text be available.

It should be pointed out, however, that studies have found even professional bibliographic indexers to disagree on a substantial proportion of categorization decisions [Fie75, HZ80, Cle91]. Indeed, whether there is really a single, correct set of decisions for a categorization problem depends on the reason for which categorization is being done.

This problem is similar to that widely faced in evaluation of text retrieval systems. For the purposes of comparing text representations the problem is less important than if we were interested in absolute levels of effectiveness of categorization systems. In addition, we can attempt to choose a test collection for which there is evidence that judgments are more consistent than usual. Nevertheless, it will be important to be aware of the problem of inconsistency in interpreting effectiveness results on text categorization.

6.4.5 Contrast with Evaluation of Text Retrieval

As mentioned earlier, evaluation for text retrieval is typically done by macroaveraging over queries. In addition, a common practice is to compute the average

precision at a number of fixed recall levels. This gives an intuitive sense of the tradeoff between recall and precision that will be observed by users as they look further and further down a ranked list of documents.

Van Rijsbergen refers to the above approach, which assumes that system effectiveness can be inferred directly from effectiveness of individual queries, as the *predictive* approach to evaluation [van81]. In contrast, the *descriptive* approach requires that the effectiveness measurements that are averaged correspond to specified system parameter settings.

The approach we have outlined above for evaluating text categorization is a descriptive approach, since we require that the system be run, with whatever parameter settings, and a full categorization be produced, before any evaluation is done. This seems appropriate for categorization systems, which will often be expected to operate autonomously. It would be misleading to average precision at fixed recall levels for a category unless the categorization system has a way of producing the same recall level on all categories. No reported categorization system can do this.

6.5 Summary

Text categorization is a task with many practical applications as well as close relationships to other text classification tasks, such as text retrieval. Large training and test sets are available for this task, and the nature of the task makes experimental control and interpretation of results relatively straightforward.

Unlike the situation in text retrieval, there are not one or two basic techniques that have become widely accepted for text categorization. However, there appears to be a broad class of statistical categorization techniques, any of which would be a reasonable vehicle for investigating text categorization. We noted that one issue requiring further research as part of any use of categorization to study text representation would be categorization with respect to multiple overlapping categories.

While no standard method has been agreed on for evaluating text categorization systems, some variant of the contingency table measures has been used in most careful evaluations. For our purposes the best approach seemed to be to compute both macroaveraged and microaveraged measures, since they communicate different things about a text representation.

In summary, text categorization is an excellent task for investigating the effect of text representation choice on content-oriented text classification, and we decided to use this task in our further investigations of text representation. In the next chapter, we describe a software system developed for carrying out such investigations.

CHAPTER 7

THE MAXCAT SYSTEM

For the CACM experiments reported in Chapter 5, an easily implemented but inefficient design was used for the text retrieval software. A number of techniques were used, such as loading the index files for an entire document collection into Lisp data structures, that would not be practical with larger document collections or with text representations containing very large numbers of terms.

A number of other factors also argued for new experimental software to be built. Different text representations, such as clusters and phrases, were handled by special purpose code in the Lisp-based retriever. It was desirable to provide a general mechanism for defining the terms in a new text representation as combinations of terms from existing representations. The switch from text retrieval to text categorization as our experimental vehicle also required some new capabilities. Finally, an efficient collection of C functions for building and accessing indexed files became available, but interfacing this to the current Lisp-based retrieval software would have been problematic.

For these reasons, the decision was made to construct a new software package, Maxcat, before continuing with our experiments on text representation. In the following section, we discuss the design of the Maxcat system and the motivations for that design. We then describe the Maxcat system in detail, and finally discuss some of our experiences with using it and opportunities for further enhancement of the system.

7.1 Design of Experimental Text Retrieval Systems

The lack of long-term research projects studying text categorization, as opposed to long-term operational projects and short-term research projects, has meant that there is little literature on the design of environments for experiments on text categorization. However, some literature exists on systems for experimenting with text retrieval [NKM77, BKZ82, BK82, CR82], particularly for the long-running SMART project [SL65, Sal71, Fox83, Buc85, FK88].

Fox [Fox83] describes the SMART system as operating in three basic stages: indexing, search, and evaluation. Our interest in exploring variations on text representations suggested that we consider the processes occurring in the indexing stage of a text retrieval system in more detail. The fact that Maxcat's main purpose is text catgorization rather than text retrieval, and the potential for applying a wide range of machine learning algorithms to this task, suggested breaking down the search stage of the SMART model into several stages as well. We also looked

at published experiments on machine learning, particularly studies that compared two or more techniques in a classification system. Examination of where in the classification process the techniques intervened suggested where divisions in the classification process should be made for maximum flexibility.

The result was the following model of processing for Maxcat. We indicate in italics correspondences with the SMART model. Most of the processes mentioned occur in SMART as well, but making them explicit aided in finding an appropriate decomposition for Maxcat:

1. Sample Formatting (*Indexing*): Separation of text stream into documents, and assignment of a unique ID number to each document.

2. Feature Extraction (*Indexing*): Detection of regularities in raw text or in feature / document / value triples, driving the creation or extension of a set of features, each feature being assigned a unique ID number.

3. Feature Valuation (*Indexing*): Finding the value of each of a set of features for each document in a corpus.

4. File Organization (*Indexing*): Creating indexed files for access to feature / document / value triples.

5. Pattern Collection (*Indexing/Evaluation*): Creating one or more partitions of documents (which may come from one or more corpora) into training and test sets.

6. Feature Collection (*Indexing*): Creating a set of potential predictor features from the union of one or more existing feature sets.

7. Category Collection (*Indexing/Evaluation*): Choosing a set of categories to act as criterion features, i.e. features whose values the system will attempt to predict on test documents.

8. Feature Judging (*Indexing/Search*): Analysis of the properties of a set of predictor features with respect to a set of training documents and a set of criterion features.

9. Feature Selection (*Indexing/Search*): Selection, typically based on feature judgments, of the subset of potential predictor features to use for learning.

10. Training (*Search*): Creation of one or more classification functions from the training set of documents.

11. Post-Training (*Indexing/Search*): Pruning or other modification of classification functions.

12. Scoring (*Search*): Use of classification functions to compute a score for each category/test document pair.

13. Categorization (*Search*): Conversion of category/test document scores into binary categorizations.

14. Evaluation (*Evaluation*): Evaluation of effectiveness of categorization, either by comparison to a specification of correct categorization, or by indirect means.

Note that while we discuss the above processes as they are manifested in text categorization experiments, the above breakdown is actually applicable to many types of experiments in machine learning. The Maxcat system could be used for experiments on medical diagnosis, plant classification, or any of thousands of other areas in which inductive learning has been used. The system can, of course, also be used for traditional text retrieval experiments based on request sets. Not all stages in the above breakdown are necessary for all experiments.

7.2 The Structure of Maxcat

The design considerations listed in the previous section, as well as general principles of abstraction and modularity, led to a five-layer structure for the Maxcat system:

1. Keyed file package

2. Sparse matrix access and creation

3. Matrix mathematics

4. Machine learning

5. Shell scripts for experiments

In addition, a number of indexing programs, which were viewed as falling outside the Maxcat system, were built. We first discuss these, and then discuss each of the Maxcat layers in turn in the following subsections. The Maxcat system runs under UNIX and uses a number of UNIX utilities.[1]

7.2.1 Indexing Programs

Before discussing the Maxcat code, we should briefly mention the software used in the Sample Formatting, Feature Extraction, and Feature Valuation stages. This code must deal with idiosyncracies of each document collection and of the particular kind of feature (word, bracketed phrase from tagged corpus, and so on) being extracted. Furthermore, the Maxcat system can be applied to other machine learning problems, where features may extracted in very different ways.

[1]UNIX is a trademark of Bell Laboratories.

Such software is viewed as lying outside the Maxcat system, which assumes its inputs will be feature/pattern/value triples.

Rather than attempting to produce general purpose software for these stages, one-shot C, *awk*, or *perl* programs and shell scripts were used.[2] More generally useful indexing programs are, of course, possible, but construction of such programs was not a priority for our experiments.

All indexing programs produced as their outputs a *dictionary file*, a *transaction file*, or both. The nature of these files is described below.

7.2.1.1 Dictionary Files

A dictionary file specifies a feature set by listing a unique identifier for each feature, along with information useful in determining what value this feature takes on for a document. For instance, this is the first few lines of a dictionary file for a feature set of words with their original capitalization retained:

```
3 1 COMMERZBANK
3 2 SEES
3 3 LOWER
3 4 OPERATING
3 5 PROFIT
3 6 THIS
3 7 YEAR
3 8 FRANKFURT
3 9 April
3 10 1
3 11 Commerzbank
3 12 AG
3 13 CBKG
3 14 F
3 15 management
3 16 board
3 17 chairman
3 18 Walter
   :
   :
```

Each entry consists of a two part ID (see the next section for more on IDs) followed by information useful in computing the feature value. In the above case, the information provided in a string of characters to look for in text. Note that the dictionary might not, however, uniquely define the value to be given to a feature for a document. The above dictionary might be used with several different indexing

[2]Most of these programs were built by Stephen Harding at the University of Massachusetts and Peter Shoemaker at the University of Chicago.

programs for scanning text. One might output binary values for presence or absence of words, another might count the number of occurrences of each word, and a third might output both number of occurrences and position of each occurrence.

As another example, the following dictionary specifies that the value of each feature in feature set 9 is to be found by adding up the values of one or more features from feature set 7:

```
9 1 but/CC  7 1 1.0  7 27 1.0  7 234 1.0
9 2 he/PPS  7 2 1.0  7 13 1.0  7 3831 1.0
9 3 left/PPS  7 3 1.0  7 173 1.0
9 4 that/CS  7 5 1.0  7 383 1.0  7 9487 1.0
9 5 left/JJ 7 43 1.0 7 18338 1.0
9 6 kalamazoo/NN 7 44 1.0
   :
   :
```

The above dictionary is the kind that would be used with the *ero* program (see Section 7.2.4). Here the string form of the feature that immediately follows the two part numeric ID is present solely to enable a human to interpret the dictionary file more easily. It is not used in finding feature values. This dictionary happens to be for syntactically tagged words, collapsed on capitalization but not on syntactic class.

7.2.1.2 Transaction Files

The second form of file produced by indexing programs is a *transaction file.* A transaction file includes a record for each feature ID/document ID pair where the feature takes on a nondefault value for that document. The following transaction file:

```
2 1 1
2 1 15
2 1 30
2 1 78
2 1 83
2 1 84
2 1 87
  :
  :
2 1 21636
2 1 21736
2 2 1
2 2 4
2 2 5
2 2 17
  :
  :
```

specifies, for instance, that feature (2,1) has a nondefault value in documents with IDs 1, 15, 30, ..., 21736. No feature value is included in the transactions, because these are binary features where the nondefault value is always 1. The set ID is omitted from the two part document ID because it is the same for every transaction. (This example file happens to specify occurrences of controlled vocabulary indexing categories in documents.)

The following transaction file is for occurrences of words in a corpus:

```
4 1 1 1 24 1
      :
      :
4 1 98 1 58 1
4 1 99 1 8 1
4 1 100 1 4 1
      :
      :
4 17 98 1 58 1
4 17 100 1 4 1
5 1 1 7 24 2 41 57 182 217 253 280
5 1 4 2 85 48 150
5 1 5 1 3 38
5 1 6 3 32 2 170 215
5 1 11 1 5 73
      :
      :
```

The format of these transactions is:

<f-set ID> <f-indiv ID> <d-indiv ID> <wdf> <maxwdf> <pos>+

where the fields are defined as follows:

- < f-set ID >: The set ID corresponding to the feature for which this transaction gives a value.
- < f-indiv ID >: The individual ID corresponding to the feature for which this transaction gives a value.
- < d-indiv ID >: The individual ID corresponding to the document for which this transaction gives a feature value.
- < wdf >: The number of times the string corresponding to the feature was found in the document assigned the specified ID, i.e. the within document frequency.
- < maxwdf >: The maximum value of the wdf field over all transactions containing this particular < d-set ID >, < d-indiv ID > pair.

- < pos >: An indication of one of the positions in the document at which the string corresponding to the feature was found. The first term in a document has position 1 and positions are assigned consecutively after that.

The above format was generated by the indexing programs used in our categorization experiments, though we did not make use of the positional information in the experiments reported in this dissertation.

7.2.2 Keyed File Package

This is a collection of C routines written by Howard Turtle and, for instance, used in implementing large Bayesian inference networks [Tur90]. It allows arbitrary record structures to be stored and retrieved using keys of up to 82 characters. Efficient access is provided using hashing on keys and a B-tree index structure. As described below, we used this package for implementing sparse matrix files.

7.2.3 Sparse Matrix Access and Creation

Most text representations, and certainly those feature sets that are extracted fairly directly from natural language text, are sparse. For instance, a significant fraction of the distinct words that appear in a large corpus will appear exactly once. If each word corresponds to an indexing term, and the value of such a term for a document depends solely on the number of occurrences of the word in the document, then the term will take on the same value for almost all documents. This default value is often, but not always, treated as 0.

If we consider the set of term/document/value triples to form a matrix, and the default value for terms to be 0, then this matrix will be *sparse*. Sparse matrices are widely used in IR, but are usually described in terms of a particular external storage implementation, the *inverted file* ([van79], p. 72; [SM83], Ch. 2). An inverted file has one record for each indexing term, listing the documents that that term occurs in, and possibly information on number or position of occurrences as well.

Mathematical algorithms for manipulating sparse matrices have been the subject of considerable attention [Pis84]. We considered using an existing sparse matrix package in Maxcat, but were dissuaded by our desire to allow default values other than 0. In addition, we could not assume that even a compact representation of an entire matrix would fit in memory at once, as sparse matrix packages usually assume. Finally, we wished to support a number of operations and variants on operations not available in sparse matrix packages. However, we did make use of certain standard sparse matrix algorithms in writing Maxcat.

We also considered using a conventional statistical package as a foundation for Maxcat. These packages support a wide range of statistical analysis and machine learning techniques, but typically do not support sufficiently efficient and flexible storage of sparse data.

Therefore, on top of Turtle's keyed file package, we implemented a set of routines for creating and accessing sparse matrices. We extended the definition of sparseness

by allowing a nonzero default to be specified for each matrix row. Each row of the matrix is stored under a separate key using the keyed file package.

The particulars of the sparse matrix package were motivated by its intended use in text classification experiments. We wanted to be able to manipulate text representations (feature sets) in a variety of ways, including analyzing feature quality, computing various statistical properties of feature sets, and producing new feature sets from subsets or unions of other feature sets. This argued for representing all features (words, phrases, clusters, and others) in a uniform manner, so the same programs could be applied to all. Similarly, we wanted to be able to define training and test sets of documents, and to allow the option of clustering documents or statistically analyzing properties of document sets.

To accomodate these needs, the idea of an *ID set* was made central to Maxcat. A set of features, or a set of documents, is represented by a set of IDs. Each ID consists of a pair of integers between 1 and $2^{32} - 1$, the first being the *set ID*, and the second being the *individual ID*. The ID 0,0 is reserved for use as the constant 1.0 when polynomials are represented.

Each column of a matrix corresponds to a single ID. There are two classes of matrices, *associations* and *archives*. Rows in associations also correspond to a single ID, but rows in archives are indexed by arbitrary keys. The two types of matrices are used for different purposes. Once an association is built, it typically is not updated. In contrast, rows can be added to archives freely and can be given mnemonic keys. On the other hand, submatrix extraction and transposition can currently only be performed on associations, since the programs for these operations take advantage of both rows and columns being indexed on a sorted set of two part numeric IDs. Archives are used mostly for storing rows that contain information on a set of IDs, such as term weights or specifications of feature subsets.

As well as containing a record for each row, a matrix file contains records listing the set of all column IDs, and all row IDs for associations. It also contains a header with summary information on the matrix, which is updated whenever a row is written to the matrix.

Standard subroutines are provided for reading, building, and writing rows. Values in the rows are accessed through functions supporting both linear and binary search. These functions hide from the client program whether a value is stored explicitly or as a default.

Besides the standard support functions, a number of standalone programs are provided for creating and manipulating sparse matrices:

- *trn-read2.c*: Reads a transaction file and two dictionary files and creates an association matrix.
- *arch-read2.c*: Reads a transaction file and a dictionary file and creates an archive matrix.
- *trn-write1.c*: Writes a transaction file for an association matrix.
- *row-write1.c*: Writes a transaction file, without row keys, for a single row of an arbitrary matrix.

- *xpl-filter.c*: Produces a new version of a matrix where only the elements specified by a second matrix remain explicit. All others are replaced by a specified default.

- *dxthresh.c*: Replaces all default values in a matrix with a specified value and all nondefaults with a different specified value.

- *dnm3.c, dnm3b.c, dnm3c.c*: Programs for printing the contents of matrices with a small number of columns (*dnm* stands for "display narrow matrix"). These are useful when testing programs.

- *transpose-mx1* and variants: These C shell scripts form the transpose of association matrices by using *trn-write1*, UNIX *sort*, and *trn-read2*. The *-xpl* versions assume that all default elements of the matrix have the value 0.0, which can provide a large savings in both space and time. The *-1def* versions are slightly less efficient than the *-xpl* versions, but allow any transposition when any single value is the default for all rows. The *f-* versions assume that all row IDs have the same set id, and all column IDs have the same set ID. This provides some speedup in transposing.

- *mxtract.c*: Extracts a submatrix from a matrix, where the row and column IDs to be extracted are specified via rows stored in appropriate archives.

- *build-test-archives.c, build-test-matrices.c*: These build a variety of matrix files with various properties for use in testing. Several other programs that build matrices to test particular programs have also been written.

7.2.4 Matrix Mathematics

Storing associations among terms and documents as sparse matrices let us cast many of the mathematical operations to be performed in these experiments as matrix math operations. Our strategy, whenever possible, was to avoid writing a new C program or even modifying an old one to implement a computation. Instead, a shell script making calls on one or more of a set of very general matrix math programs was used. This had the effect of trading space and time efficiency (sometimes in large amounts) for being able to implement new techniques quickly and with a high degree of reliability.

The matrix math programs used are listed below. Some correspond to programs that typically appear in sparse matrix packages, while others are more unusual and are motivated by the particular needs of machine learning or IR experiments.

A number of the programs make use of a small library of subroutines for unary, binary, and k-ary mathematical operations. Again some of these are fairly standard math operations (square root, addition, and mean of k elements for instance) while others are motivated by the particular needs of our experiments (entropy, threshold first argument against second, count number of nonzeros in k elements).

- Matrix Multiplication and Conventional Matrix Algebra

- *ero.c*: Takes as input an association matrix and a file specifying an output matrix in terms of elementary row operations on the input matrix, and produces the output matrix.
- *mmrc2.c*: Matrix multiplication, under the assumption that the physical rows of the second matrix correspond to its logical columns. The case where the first matrix is a classification matrix, i.e. is viewed as consisting of linear polynomials, is handled specially.
- *mmrc3.c*: A variant of *mmrc2* that achieves additional speed in several ways, including assuming that all column IDs come from the same ID set.
- *mmrr2.c*: Matrix multiplication, under the assumption that the physical rows of the second matrix correspond to its logical rows. This uses a variant on the Gustavson algorithm ([Pis84], pp. 253–258).
- *outer.c*: Takes a row as input and produces its outer product, a matrix.

• Elementwise Math

- *metm.c* (and *retr.c*): Applies a binary operation to every element of a matrix (row) in combination with a command line argument, producing an output matrix (row).
- *mmtm.c* (and *rrtr.c*): Applies a binary operation to every element of a matrix (row) in combination with the corresponding element from another matrix (row), producing an output matrix (row).
- *mrtm.c*: Applies a binary operation to every element of a matrix in combination with the element having the same column in a specified row. Produces an output matrix.
- *mte.c* (and *rte.c*): Applies a k-ary operation to all elements of a matrix (row), outputting a single value.
- *mtr.c*: Applies a k-ary operation to the elements of each row of a matrix, outputting a row containing these values.
- *mtm.c* (and *rtr.c*): Applies a unary operation to each element of a matrix (row) outputting a matrix (row).

• Miscellaneous Programs

- *diag.c*: Produces a diagonal matrix with the same rows and columns as an input association matrix.
- *max-xpl.c*: Takes an input matrix and outputs a new matrix retaining only the top k explicit values from each row.
- *msmy.c*: Outputs one of a number of single values, specified by a command line argument, such as number of rows in matrix, number of columns, or sum over all values in the matrix.
- *thresh-row.c*: Thresholding of a single row against a value, outputting a row. A number of variants are available.

7.2.5 Machine Learning

A variety of techniques for creating and processing features, inducing and applying categorization functions, and evaluating categorizations were needed for the experiments. Where possible, these were implemented as shell scripts using the matrix math programs. Otherwise, specific C programs were written.

Our model for how learning algorithms should be implemented evolved during the course of the disssertation experiments, and is still evolving. Our first learning program, which implemented the Bayesian algorithm in [Mar61], computed all probability estimates directly from raw data. The output of the program was a matrix of linear classifiers, which could be applied to any test data matrix.

When we implemented a more sophisticated Bayesian classifier proposed by Fuhr [Fuh89], the model of outputting linear classifiers would not work, as Fuhr's formula does not result in a linear classifier. Therefore, the program *fuhr3* is given both training and test data as inputs and uses the training data in scoring test data. At the same time, to increase our flexibility in experimentation, probability estimation was removed and embodied in separate programs. (Except of course for the categorization probabilities that the program produces as its output.) This enables the same algorithm to be used with a wide variety of probability estimates produced by other programs.

We classify the machine learning programs into the following rough categories, but a number of them actually have several purposes.

- Feature Extraction and Judging
  - *capcollapse1*: A script that produces an dictionary file, for use with *ero*, specifying that the within document frequencies for all features that vary only in capitalization should be added to produce the values for a combined ID.
  - *cosine*: A script that produces a similarity matrix for a set of row IDs using the cosine correlation. Produces large, non-sparse matrices.
  - *nncosine.c*: Produces the nearest neighbors for each of a set of row IDs as defined by the cosine correlation.
  - *fnn*: A script that produces the nearest neighbor graph for a set of IDs, given a similarity matrix.
  - *info-gain*: Estimates information gain (system mutual information) for each pair (Wi,Cj) drawn from two sets of IDs.
  - *rnnc.c*: Produces an *ero* dictionary file for reciprocal nearest neighbor clusters of a set of IDs, given a matrix containing the nearest neighbor graph for the IDs.
  - *stopremove*: Produces a new dictionary file by removing from an existing dictionary file all IDs corresponding to a stopword list.
- Frequency and Probability Estimation

- *cooc-pair-cond*, *cooc-pair-dist*, *cooc-pat-cond*, *cooc-pat-count*: These produce, under various assumptions, a matrix of estimates of the number of trials resulting in the event $(W_i = 1, C_j = 1)$ for all pairs of row and column IDs.
- *mat-ele*, *mat-mle*, *mat-mme*: Estimates, under expected likelihood, maximum likelihood, and minimax estimators respectively, of the probability of the event $(W_i = 1, C_j = 1)$ for all pairs of row and column IDs.
- *norm-scores-1*: Normalizes values in each row of a matrix to have mean 0 and variance 1.
- *pair-cond*, *pair-dist*, *pat-cond*, *pat-count*: These produce, under various assumptions, a row of estimates of the number of trials resulting in the event $C_j = 1$ for each column ID.
- *row-ele*, *row-mle*, *row-mme*: Estimates, under expected likelihood, maximum likelihood, and minimax estimators respectively, the probability of the event $C_j = 1$ for all column ids.

• Information Retrieval Estimation Methods

- *norm-wdf*, *norm-wdf-2*: Produces, from a within document frequency matrix, a matrix of normalized WDF values.
- *scaled-idf*: Produces, from a binary term occurrence matrix, a row of IDF values scaled to lie between 0 and 1.
- *turtle*: A method of computing term significance weights for documents.

• Learning

- *tr-maron-2.c*: Outputs a matrix of linear classifiers, one for each category, using a Bayesian formula suggested by Maron [Mar61]. The *mmrc2* or *mmrc3* programs can be used to apply the classifiers to a set of test documents to produce scores for the categories.
- *fuhr2.c*, *fuhr3.c*: Compute a score for each of a set of categories on each of a set of test documents, using a Bayesian formula suggested by Fuhr [Fuh89]. Uses matrices with the necessary probability estimates as inputs.

• Evaluation

- *ebc0*: Produces microaveraged evaluation measures by comparing a transaction file of categorizations with a transaction file of correct categorizations.
- *ebc1.c*: Produces microaveraged, macroaveraged, and other evaluation measures from a matrix of category assignments and a matrix of correct category assignments. Provides per category evaluation scores as well.

- *eval-format.c*: Translates a matrix of ranks of relevant documents into a form usable by *EVAL*, a program used by the University of Massachusetts IR Laboratory for producing recall precision tables for text retrieval experiments.
- *rank-columns.c*: Replaces each value in a row with its rank among the other values in its row.
- *threshtrn1*, *topkn-trn*, *topkn-trn-sub*: Prepocessing of transaction files for ebc0.

7.2.6 Experiment Scripts

A typical experiment involves twenty to one hundred or more calls on various of the above mentioned programs. In order to manage this complexity, each experiment is implemented as a set of shell scripts.[3] A file defining symbolic names for all directories in the hierarchies of files used in this study is loaded at the front of each script. By redefining the top level directory in this file, the entire set of scripts can be run on a set of test data before the actual run, and this was done before each primary experiment. Using a small test set was invaluable in quickly detecting bugs, as runs on the complete data set often took several days.

7.3 Experience and Directions for Improvement

The highly modularized approach taken in the Maxcat system worked well. It was possible to experiment with changes in a variety of system parameters quickly and without modifying existing code. Since intermediate results were written out at a number of stages, many computations (particularly probability estimates) only had to be run once, and could be reused in a number of experiments, sometimes in ways unanticipated at the time of their computation. The constant writing and reading of intermediate results undoubtedly increased computation time for any particular set of calculations over what would be required by a monolithic program. But the total computation time for a large number of experiments was greatly reduced. Furthermore, the risk of undetected programming errors when testing a new strategy was reduced.

The primary drawback of the approach taken was the disk space required for intermediate files. This was alleviated to some degree by the use of the UNIX *compress* utility and by including calls to delete less useful intermediate files in the shell scripts. A tradeoff sometimes had to be made between the space required for a file and the time necessary to recompute it if needed again, as well as the additional

[3]We attempted at one point to characterize the dependencies between intermediate results using the UNIX *make* utility. The complexity of the dependencies, the difficulty of specifying correct *make* scripts when files are spread across a number of directories, and the problem of files that are updated (such as archives), resulted in this attempt being unsuccessful and our returning to using shell scripts.

steps that would need to be included in scripts for future experiments. Emphasis was placed on retaining files that were expensive to compute and files that came logically early in computations, such as basic probability estimates.

While we originally intended transaction files only to be used as the initial input for building matrices, we found this representation to be quite useful for other purposes as well. For instance, we originally built a program for transposing matrices that operated directly on the indexed form of a matrix. However, experimentation showed that it was much faster to write out the matrix as a transaction file, sort it on the original column key, and build the transposed matrix from the sorted transaction file. It is likely that replacing *mxtract* by a transaction-based method would result in a similar improvement, though we did not explore this. Transaction files were also useful for doing quick analyses of the contents of a matrix, particularly before we constructed programs such as *mtr*. Our first evaluation program, *ebc0*, was a shell script using a number of UNIX utilities to operate on transaction files.

The disadvantage of the transaction files used was that each feature/document pair that had a nonzero value required a record. While large numbers of zeroes are common for the initial features produced from text, zeroes are rare in most of the other matrices used in experiments. It would be relatively straightforward to adapt transaction files to handle nonzero defaults, and only slightly less straightforward to handle row-specific or column-specific defaults.

While the use of nonzero defaults and row-specific defaults in our sparse matrices saved considerable space, there is clearly room for improvement in our storage scheme. The inclusion of column-specific defaults would avoid matrices suddenly increasing greatly in size when transposed. Beyond this, the ability to specify the default value of a matrix element as a simple function of the defaults for both its row and column might be useful, though at that point it may be better to give up and simply require that the value be computed by the user program rather than by the matrix access program.

Other possible techniques for space efficiency include using a larger radix in transaction files, and defining specialized formats for binary matrices and other matrices where stored values have a small number of significant bits. The latter is nontrivial to do correctly and portably, however. Our initial implementation of sparse matrices defined set IDs to be unsigned short integers (in the range 0 to $2^{16} - 1$) so that they would only take up 16 bits in matrix entries. When Maxcat was moved to a new workstation and a different implementation of UNIX, alignment errors with reading and writing matrix files surfaced, and set IDs had to be converted to 32 bit unsigned long integers.

Some sets of values simply cannot be stored efficiently. Sets of conditional or joint probabilities contain little redundancy, particularly when biased estimators are used to compensate for small sample errors. It will often be necessary, as it was in these experiments, to restrict the computation and storage of these values to selected subsets of IDs.

Even with various approaches to efficiency in storing matrices, at some point it will become impractical to, for instance, do mathematical computations on matrices by writing out matrix files for intermediate results. One approach to handling this,

while still allowing computations to be specified via shell scripts rather than by rewriting C code, would be to write an interpreter or compiler for matrix equations.

Finally, there is considerable room for improvement in the individual programs that operate on matrices, as well as new programs that might be added. A major target for improvement, before larger datasets are attempted, are those programs that do $O(n^2)$ or $O(mn)$ operations on pairs of rows. Currently these programs, particularly matrix multiplication, are hampered by our restriction that only one row from a matrix can be stored in memory at a time. This constraint could be relaxed considerably, while still allowing very large matrices to be processed, by doing more sophisticated buffering.

7.4 Summary

We have developed a flexible software system, Maxcat, for experimentation with text classification. Its main advantages over existing experimental IR software are in its facilities for flexibly defining new text representations in terms of old representations, and in its support for varying a large number of characteristics of machine learning techniques in text categorization. Maxcat can also be used for experiments on classification in domains not involving text. In the next chapter we describe our first experiments on text categorization using Maxcat.

CHAPTER 8

TEXT CATEGORIZATION: FIRST EXPERIMENTS

We discussed in Chapter 6 how the text categorization task provides an excellent context for comparing the quality of text representations. However, methods are less standardized for text categorization than for text retrieval, and we had not previously worked on the text categorization task. We therefore conducted a number of preliminary experiments, focusing purely on text categorization issues, before returning to our hypotheses on syntactic phrases. These experiments are reported in this chapter and the next, and use only minor variants of a single text representation, unstemmed words. The emphasis is not on the text representation but on understanding the categorization process, particularly as applied to the test collection we chose to use.

An appropriate avenue for this investigation was to replicate a classic categorization study. We chose to replicate Maron's study [Mar61], since it is widely cited and was a forerunner of much of the work on statistical categorization. Since none of this later work has actually compared new algorithms with Maron's algorithm, or, usually, with any other algorithm, we hoped also to gain some insights into the relative merits of techniques for text categorization.

This chapter begins by discussing a text categorization test collection, the Reuters-22173 collection. We then turn to Maron's original experiment, and how we replicated it on the Reuters texts. Finally we discuss the results, which motivated a number of changes in our categorization methods and experimental technique.

8.1 The Reuters Data

In several articles [HKC88, HANS90, Hay90], Hayes and colleagues have reported on CONSTRUE, a rule-based text categorization system that Carnegie Group, Inc. developed for Reuters, Ltd. CONSTRUE achieved a microaveraged recall of 89% and a microaveraged precision of 92% for 135 categories on a previously unseen test corpus of 723 documents [HW90].[1] It should be noted that microaveraged measures are heavily influenced by effectiveness on the most frequent categories, and that 13 categories account for 59.3% of category assignments on the CONSTRUE test set.

[1]The method used to compute microaveraged precision was slightly nonstandard (Hayes, personal communication). However, in our own results we found little actual difference in the values produced by the Carnegie Group formula for microaveraged precision and by the standard formula for microaveraged precision.

Nevertheless, the degree of agreement between machine and human categorization for CONSTRUE is much higher than reported in almost any other text categorization experiment, and is higher than the degree of agreement usually reported between human indexers [Bor64, Fie75, HZ80]. This suggested that the collection of training and test documents used by Carnegie Group would make an excellent test collection for investigating text categorization and text representation.[2]

In this section we discuss the characteristics of the raw Reuters text and of the human category assignments made to it. We then discuss how we prepared this text in a fashion that would allow flexible and replicable experiments on text categorization.

8.1.1 The Raw Text

The Reuters texts were supplied by Carnegie group to the University of Massachusetts in the form of 74 text files totaling just over 31 megabytes. Fifty-four of the files contained the approximately 21,450 articles that were used by Carnegie Group developers during the construction of CONSTRUE's rulebase. Another file contained the 723 unseen test articles used in published evaluations of CONSTRUE's effectiveness. The remaining 19 files contained 5,125 stories, some of them just additional copies of the 723 test articles but most of them new, which it is believed were not used for either training or testing of CONSTRUE (Hayes, personal communication).

All articles were distributed via the Reuters newswire during 1987. Perhaps the most notable characteristic of these articles is that they are full text documents. With few exceptions, IR test collections in the past have been limited to titles, abstracts, and other summary information. The Reuters texts vary in length from single line bulletins to multipage articles. They include formatted and unformatted tables of numeric data as well as text. Content includes not only reports of recent events, but also long feature stories, quotes on market prices, and corporate earnings reports.

How many word tokens are present in a corpus depends on how words are defined and the text tokenized. However, for typical definitions of "word," the corpus contains approximately three million word occurrences, placing the Reuters corpus toward the large end of corpora used for IR or NLP research in the past, though both fields are rapidly moving toward much larger corpora.

The primary motivation for Reuters in producing these texts was real-time distribution to human readers, with long term archival storage on computer apparently a secondary consideration. While there is an intended structure to the articles, the segmentation of the text stream into documents and the segmentation of documents into fields is by no means uniform, and a number of typographical and transmission errors are present. The texts also include control codes and other nontextual information related to the original transmission of the stories.

---

[2]We thank Carnegie Group and Reuters for making this corpus available to the University of Massachusetts for experimental purposes.

A major difference between the Reuters collection and text collections based on archival document sets is the presence of expanded, modified, and corrected versions of stories. These minor variants of stories were a service for the real time users of the newswire, since they could be used or ignored as appropriate. They are considerably more problematic in the context of a long term database, and in particular for a database used in evaluating text classification systems. We refer to these slightly modified stories as *near duplicates*, and discuss the implications of them in Section 8.5.

8.1.2 The Categories

The CONSTRUE system assigned to each Reuters article zero or more categories from a set of 674 total categories. The 674 categories are broken down into five groups as follows[3]:

1. TOPICS (135) : These are subject categories of economic interest. We list the complete set of them in Figure 8.1.
2. ORGANIZATIONS (56) : Economically important regulatory, financial, and political organizations.
3. EXCHANGES (39) : Stock and commodity exchanges.
4. PLACES (175–176) : Countries of the world.
5. PEOPLE (267–269) : Political and economic leaders important at the time of the corpus construction.

One of the unusual aspects of the CONSTRUE project was that this category set was developed in parallel with the construction of the CONSTRUE system, by a team of Carnegie Group and Reuters personnel [HW90]. Definitions of categories were adapted to make mechanical assignment easier, which undoubtedly contributes to the very high levels of effectiveness that CONSTRUE is able to achieve (Hayes, personal communication). The category assignments in the Reuters corpus were performed by two Reuters journalists stationed at Carnegie Group, adding to the level of consistency. Nevertheless, there are still a significant number of errors in the raw category assignments, including mispelled categories, duplicated categories, categories placed in the wrong field (i.e. TOPICS vs. PLACES), and strings (*inflation*, *gbonds*, *tbill*) that are clearly intended to be categories but that are not part of the official category set.

---

[3]Documentation on the CONSTRUE experiments has been somewhat contradictory with respect to the exact number of categories in some groups.

ACQ	GRAIN	PLYWOOD
ALUM	GROUNDNUT	PORK-BELLY
AUSTDLR	GROUNDNUT-MEAL	POTATO
AUSTRAL	GROUNDNUT-OIL	PROPANE
BARLEY	HEAT	RAND
BFR	HK	RAPE-MEAL
BOP	HOG	RAPE-OIL
CAN	HOUSING	RAPESEED
CARCASS	INCOME	RED-BEAN
CASTOR-MEAL	INSTAL-DEBT	RESERVES
CASTOR-OIL	INTEREST	RETAIL
CASTORSEED	INVENTORIES	RICE
CITRUSPULP	IPI	RINGGIT
COCOA	IRON-STEEL	RUBBER
COCONUT	JET	RUPIAH
COCONUT-OIL	JOBS	RYE
COFFEE	L-CATTLE	SAUDRIYAL
COPPER	LEAD	SFR
COPRA-CAKE	LEI	SHIP
CORN	LIN-MEAL	SILK
CORN-OIL	LIN-OIL	SILVER
CORNGLUTENFEED	LINSEED	SINGDLR
COTTON	LIT	SKR
COTTON-MEAL	LIVESTOCK	SORGHUM
COTTON-OIL	LUMBER	SOY-MEAL
COTTONSEED	LUPIN	SOY-OIL
CPI	MEAL-FEED	SOYBEAN
CPU	MEXPESO	STG
CRUDE	MONEY-FX	STRATEGIC-METAL
CRUZADO	MONEY-SUPPLY	SUGAR
DFL	NAPHTHA	SUN-MEAL
DKR	NAT-GAS	SUN-OIL
DLR	NICKEL	SUNSEED
DMK	NKR	TAPIOCA
DRACHMA	NZDLR	TEA
EARN	OAT	TIN
ESCUDO	OILSEED	TRADE
F-CATTLE	ORANGE	TUNG
FFR	PALLADIUM	TUNG-OIL
FISHMEAL	PALM-MEAL	VEG-OIL
FLAXSEED	PALM-OIL	WHEAT
FUEL	PALMKERNEL	WOOL
GAS	PESETA	WPI
GNP	PET-CHEM	YEN
GOLD	PLATINUM	ZINC

Figure 8.1 The set of TOPICS categories used in the Reuters collection.

8.1.3 The Prepared Text

The original Reuters corpus might be called "informally formatted." The intended structure includes: a separation of the text stream into articles and the presence of fields for controlled vocabulary categories, text, and other information. The textual portion of the documents include titles, bylines, datelines, paragraphs, table headers, and other subdivisions within the text. However, since having this structure exactly right was not crucial to the human consumers of the texts, it is not surprising that the structure is not always present consistently or correctly.

To conduct categorization experiments, particularly in a replicable manner, it was necessary to produce a much more carefully formatted version of the collection. The text had to be separated into documents, and information recorded as to what controlled vocabulary categories and text representation features are present for each document. It is desirable that such a standardized form of the collection eliminate some of the anomalies of formatting in the original collection, so that the record for each document has the same structure. Our goal in doing this was to produce a version or versions of the collection on which other researchers could reproduce our results, without requiring that they used exactly the same software that we did.

On the other hand, one does not want the format produced to restrict the range of possible experiments that can be performed on the collection. For instance, a format that does not provide the original string of characters that made up the document cannot be used to study alternate segmentations of the text field into words, or alternate segmentations of the document into text and non-text.

To address the conflicting demands of standardization and flexibility, we used two levels of standardization in preparing the collection.[4] The first level, which corresponds to the Sample Formatting stage discussed in the previous chapter, involved detecting boundaries between articles, inserting a unique identifier into the text of each article, and inserting an explicit delimiter between articles. We also explicitly marked each document as belonging to the original training or test set, something which had been recorded only implicitly (by presence in a particular file) before.

Even this task involved some choices that could be debated. The boundaries between documents were not always marked in the same way and, given the size of the corpus, it was impossible to check by hand whether all documents were separated correctly. We tuned the program for separating documents by, for instance, requiring that each document contain exactly one instance of each of the controlled vocabulary field markers, since these clearly were intended to occur in each document. Despite this, the final formatted version of the collection we produced contained 21,450 training documents and 723 test documents, vs. the 21,451 training and 723 test documents reported by Carnegie Group (Hayes, personal communication). Which figure is correct is not immediately apparent. (Documentation on Carnegie Group's segmentation was unfortunately not available.)

---

[4]Steven Harding at the University of Massachusetts at Amherst, and Peter Shoemaker at the University of Chicago were responsible for much of the work on data preparation.

The original collection had come in files of widely varying sizes. In formatting the text we repackaged the formatted articles 500 to a file for convenience. Examples of the formatted text can be found in Chapter 10. We have named this formatted version of the corpus the Reuters-22173 test collection.

The second format used represents the collection as a pair of files: a dictionary file and a transaction file, as discussed in Chapter 7. Producing the dictionary file corresponds to the Feature Extraction stage of categorization, while producing the transaction file is the Feature Valuation stage. Each basic text representation, including the controlled vocabulary categories to be predicted, requires its own pair of files under this format. Since this format precisely identifies a set of terms and specifies which are present in each document, this format does not allow experiments on alternate ways of extracting features from raw text. However, many other sorts of experiments, including ones on feature selection, clustering, and categorization algorithms, can be conducted using this format. This format can also be converted directly into the indexed files necessary for efficient processing, and the completely regular nature of the format enables easy replicability of experiments.

Converting raw text into the transaction file format involves making a number of arbitrary decisions. Even in producing the transaction file for controlled vocabulary categories, the issue arose of how to handle problems such as typographical errors and categories that were placed in the wrong subfield. Our principle here was to maintain maximum flexibility for later experimentation where possible. Therefore, we treated each distinct pair of the form <controlled vocabulary subfield, string in that subfield> as a separate binary feature of documents and created the transaction file accordingly. This resulted in a dictionary file with 672 distinct entries. We later expanded this to 689 entries by adding the 17 TOPIC categories that had no occurrences on the corpus, but that were legal categories (Hayes, personal communication). By avoiding making subjective decisions about how to correct invalid entries, we enable the transaction file to be used for experiments that involve different corrections of the entries, since almost any correction would result in terms that are the union of our maximally specific terms.

Creating dictionary and transaction files for the textual portion of a document involves making a commitment to some text representation. While multiple transaction files were obviously necessary in some cases (a text representation based on phrases vs. one based on words, for instance) we tried to make the transaction files produced correspond to relatively specific analyses of the text, on top of which more general analyses could be done. So, for instance, capitalization differences and syntactic tags were preserved, no stop words were omitted, and no stemming was done in our transaction files for words. This retained the ability to experiment with a variety of methods for collapsing or not collapsing words based on capitalization, morphological, or syntactic clues, or removing words based on stopword lists, without having to redo the tedious process of building a transaction file from the text. Having described the test collection to be used for our experiments, we now turn to the original experiment we chose to replicate.

8.2 Maron's Categorization Experiment

Maron conducted one of the earliest studies of statistical methods for text categorization [Mar61]. He used a Bayesian formula to estimate the probability that a category was assigned to a document, given the presence of selected words in the document. The formula used (changing Maron's notation slightly) was

$$P(C_j = 1|D_m) = k \times P(C_j = 1) \times \prod_i P(W_i = 1|C_j = 1)$$

Here $P(C_j = 1|D_m)$ is the probability that category C_j should be assigned to document D_m. $P(C_j = 1)$ is the prior probability of assignment of C_j and is estimated by

$$\frac{N(C_j = 1)}{\sum_k N(C_k = 1)}$$

where $N(C_j = 1)$ is the number of assignments of category C_j to training documents. Since each category was assigned 0 or 1 times to each document, this is simply the number of documents to which C_j was assigned. $P(W_i = 1|C_j = 1)$ is the conditional probability that word W_i is assigned to a document given that category C_j is assigned to the document. This is estimated by

$$\frac{N(W_i = 1, C_j = 1)}{\sum_i N(W_i = 1, C_j = 1)}$$

where $N(W_i = 1, C_j = 1)$ is the number of occurrences of W_i in documents to which C_j is assigned. Since Maron treated assignments of words as binary, this is simply the number of documents in which both W_i and C_j appear. When $N(W_i = 1, C_j = 1)$ was 0, an estimate of 0.001 was used for $P(W_i = 1|C_j = 1)$ to avoid conditional probabilities of 0.

The value k was a constant chosen so that:

$$\sum_j P(C_j = 1|D_m) = 1.$$

This value subsumes the denominator in the Bayesian formula (see Section 9.2.2) on which Maron's technique is based, as well as incorporating an ad hoc scaling factor. Maron's formula assumes independence of assignment of words, given assignment of a category, and also assumes that category assignments are mutually exclusive and exhaustive.

Training was done on 260 abstracts of computer science articles. A total of 3,263 distinct words manifested somewhat over 20,000 binary word occurrences in these abstracts. A feature set containing 90 of these words was chosen by hand, though initial pruning was done by automatic criteria, including omission of high frequency function words and omission of words occurring in two or fewer documents. A set of 32 categories were created by Maron and a colleague, who together assigned them to the training and test documents.

Maron's categorization software computed estimates of $P(C_j = 1|D_m)$ for both the 260 training documents and a previously unexamined set of 145 test documents. The estimates of $P(C_j = 1|D_m)$ were used to produce a ranking of the 32 categories for each document, and Maron's evaluation was based on this ranking.

Evaluation was the weakest part of the study. Maron reported results only on the 85 test documents with two or more clue words, leaving the reader to guess at the (presumedly poor) effectiveness of the system on the other 60 test documents. Maron also resorted to ad hoc methods to deal with the fact that some documents had only one correct category, while others had multiple correct categories. For instance, Maron says that of the 66 documents with exactly one correct category, that correct category received the highest rank in 33 cases. For the 16 documents with two correct categories, he reports that in 5 of the 16 cases the correct two categories appeared at the top two ranks. This begs the question, however, of how an automatic categorization system would know how many categories to which a document should be assigned. This avoidance of the problem of multiple category assignment has also characterized much of the work on statistical text categorization subsequent to Maron's.

8.3 Replicating Maron's Study on the Reuters Corpus

We attempted to replicate Maron's study as closely as possible on the Reuters corpus. The basic model used in Maron's study—predicting binary indexing categories using binary features corresponding to words—applied cleanly to the Reuters data. Some judgment calls arose, however, particularly due to our desire to avoid any form of human intervention in the categorization process. We discuss these issues below.

8.3.1 Binary Categorization

The most important place where Maron intervened in the categorization process was in going from an estimate of the probability $P(C_j = 1|D_m)$ to a decision either to assign or not assign C_j to a document with description D_m. As described in the previous section, this was done by Maron with prior knowledge of how many categories the document should be assigned, something that is not appropriate in evaluating an automatic system.

For our experiments we required that the binary categorization be done without human intervention. In this chapter we present results comparing five fully automatic categorization methods.

8.3.2 Feature Selection

Maron selected his 90 predictor features by hand, and it is unclear how much his results depend on his skill in feature selection. We do know that words occurring in 2 or fewer documents were omitted, as well as high frequency "logical type" words, and certain other high frequency content words. He also specifies [Mar61] that the following procedure was then applied to the remaining words:

> A listing was made showing the number of times each of these 1,000 words occurred in those documents belonging to category 1, category 2, etc. In this way, one could scan the list and for each word see whether or not it "peaked" in any of the 32 categories. If a word did peak it was felt that the word would be a good clue. If the distribution was flat for a given word (i.e. it did not have a peak in at least one category), then it was rejected as a good clue.

Maron unfortunately does not define what he means by peaking. We chose to implement this restriction by selecting only words that had at least three different cooccurrence counts with categories. Since almost any word has a zero cooccurrence count with some category, this in effect required that the word have two different nonzero cooccurrence counts with categories.

Maron arbitrarily used 90 predictor features. Since we had two orders of magnitude more training data than he, it seemed reasonable to use more predictor features. Fuhr [FB90] has suggested that 50-100 samples per parameter are appropriate for

estimation in text classification problems. Taking the middle of this range we chose to use 21,450 training documents divided by 75 training documents per feature, or 286 features.

Maron did not publish his list of high frequency function words to be omitted, so we used the current stop word list from the Information Retrieval Laboratory at the University of Massachusetts at Amherst. It is reproduced here in Figures 8.2 and 8.3. Maron also screened out certain high frequency content words by hand, and we simply omitted this step of Maron's procedure.

As it turned out, more than 286 words met Maron's restrictions, both with and without the use of the stop list. Of all the words meeting Maron's restrictions, we therefore chose the 286 with the highest document frequency, so as to have features for which conditional probability estimates would be as accurate as possible, and that were relatively likely to occur in test documents.

8.3.3 Estimation Formula

Maron used a scaling factor k so that his estimates would have the property:

$$\sum_j P(C_j = 1|D_m) = 1.$$

This assumes that there is only one category per document, which was true neither for Maron's data, nor ours. This constant also subsumes a division by $P(W_i = 1)$ that is required by Maron's formula. Eliminating the constant k and making this division explicit gives the formula

$$P(C_j = 1|D_m) = P(C_j = 1) \times \prod_{i_m} \frac{P(W_{i_m} = 1|C_j = 1)}{P(W_{i_m} = 1)}$$

where i_m ranges over those words W_i for which $W_i = 1$ in D_m.

Note that this form makes explicit an additional independence assumption on the probabilities of word occurrences. Some of our experiments were conducted using the probabilities directly produced by this formula.

We also investigated using Maron's scaling technique. A difficulty arose from the fact that we had a much larger training sample than Maron's, as well as a larger number of features. These two factors combined to cause the above formula to produce extremely small or (due to the violation of independence assumptions) extremely large numbers as probability estimates. For instance, in one experiment the estimated probabilities ranged from 10^{-227} to 10^{14}, the latter of course completely impossible for a probability. Without writing special purpose math software, the best way to represent such probabilities accurately was as logarithms, which all our software did.

However, the scaling factor for document D_m,

$$k_m = \frac{1}{\sum_j P(C_j = 1|D_m)}$$

cannot be computed without converting from the logarithm of $P(C_j = 1|D_m)$, which we had stored, to the actual $P(C_j = 1|D_m)$ values. To avoid the inaccuracies

that would result from this procedure, we instead noted that, for each document D_m, the highest single value of $P(C_j = 1|D_m)$ usually accounted for the most of the sum $\sum_j P(C_j = 1|D_m)$, often by several orders of magntitude. We therefore approximated k_m by

$$k'_m = \frac{1}{P(C_g = 1|D_m)^{0.99}}$$

Here $P(C_g = 1|D_m)$ is the highest value of $P(C_j = 1|D_m)$. The logarithm of k'_m then is

$$\log k'_m = -0.99 \log P(C_g = 1|D_m)$$

which was computable without leaving the log domain. The factor of 0.99 was used in order to retain some difference among the highest probabilities estimated across the documents.

The impact of these various methods on the various binary categorization strategies we tried is discussed in Section 8.5.

8.3.4 Choice of Categories to Predict

Maron used a set of 32 invented categories for his experiment. We had available in the Reuters data a total of 674 categories which might be used. Of these, the set of 135 TOPIC categories seemed the most appropriate for our experiments, since the other classes of categories are based heavily on the occurrence of particular proper nouns in text, rather than on the general content of the text. Of the 135 TOPIC categories listed in documentation from Carnegie Group, we found that 118 actually had one or more occurrences on the Reuters corpus, so we chose this set to use.

8.3.5 Zero Probabilities

When a clue word and category did not cooccur in any documents, Maron used 0.001 as an estimate of $P(W_i = 1|C_j = 1)$. This was presented as a means to avoid estimates of 0 for $P(C_j = 1|D_m)$ without disturbing the order of $P(C_j = 1|D_m)$ values as j varied. Given 21,450 training documents instead of 260, a lower value was clearly necessary to stay true to the spirit of Maron's strategy. (We discuss a less ad hoc way of dealing with 0 probabilities in the next chapter.)

The event space that Maron used a value of 0.001 as an estimate for an event in was the number of pairs of the form $(W_i = 1, C_j = 1)$ summed over all W_i for a particular C_j. One thing we can note is that this quantity is highly variable between C_j's. We can estimate the maximum size of the event spaces by noting that no category was applied to more than 37 documents in Maron's data. Also, only four documents had 8 clue words, nine documents had 7 clue words, nineteen documents had 6 clue words, and forty-six documents had 5 clue words. So the absolute maximum size of an event space was $4 \times 8 + 9 \times 7 + 19 \times 6 + 5 \times 5 = 234$ events, and most must have been much smaller than this. If we note that the mean number of clue word assignments was 3.6 and guess that the average number of documents to which a category was assigned was perhaps 10, this gives an event space of around 36 events.

So the probability Maron used, 0.001, was a factor of 25 to 30 less than the minimum conditional probability that could be estimated from his data by maximum likelihood estimation. This is in keeping with his comment that this estimate was used to keep the probability estimate from going to zero without disturbing the ordering of categories. As an ad hoc way of duplicating Maron's strategy we used an estimate of 4% of the reciprocal of the mean sample space size as an estimate.

For our data, there were a total of 555,735 nonzero cooccurrences of 118 controlled vocabulary categories and 286 predictor words (in the no stop word list condition). This yields an estimate of $.04 \times (1/(555,735/118)) = .0000085$ for $P(W_i = 1|C_j = 1)$ when W_i and C_j have no cooccurrences. For the condition where a stopword list was used, there were 347,708 cooccurrences, giving a value of 0.000014.

8.4 Results

Maron's method, with the modifications described above, was used to estimate the probability that each of the 118 selected categories occurred on each of the 723 test documents, for a total of 85,314 probability estimates. The obvious strategy for converting these estimates to a binary categorization was *probability thresholding*, i.e. setting a threshold on the minimum estimated probability, and making all binary category assignments where the estimated probability of assignment exceeds the specified threshold. No scaling of the probability estimates was done. Table 8.1 shows the results for this run when stop words were not excluded, and Table 8.2 shows results when stop words were excluded. We chose the probability thresholds so that the ratio between category assignments and numbers of documents on the test set was some constant factor of the corresponding ratio on the training set. The ratio on the training set was 0.64 category assignments per document. So, for instance, in Table 8.1 the pairing of the proportion 0.20 with the log probability 16.7 means that 16.7 is the highest threshold such that at least $0.20 \times 0.64 \times 723$ (rounded to 93) category assignments would be made to the 723 documents in the test set.

One thing that the probability values make clear is how thoroughly the independence assumptions on which Maron's formula is based are violated. The value of 16.7 mentioned above means that the formula used is estimating 93 probabilities of $e^{16.7} = 1.8 \times 10^7$ or more, when, of course, any estimate of a probability greater than 1.0 is clearly wrong. This fact was hidden by the scaling factor in Maron's original formula, but we make it explicit here to drive home the problems with models that assume term independence. That said, all the models used in this study will assume term independence, since independence models have repeatedly proven effective in IR, and little is known about appropriate dependence models for IR.

For each threshold setting we provide both microaveraged and macroaveraged recall and precision figures. We adopt the approach of excluding 0/0 values in macroaveraging.

Since we wanted to investigate a variety of category assignment schemes, it was not possible to construct a single evaluation program that would compute, for

instance, the precision values at fixed recall intervals. This makes it difficult to compute a single quality rating, such as average precision across 10 fixed recall levels.

One reasonable single measure of effectiveness is the breakeven point between recall and precision. When none of the parameter settings explored produced equal recall and precision, we find an estimate of this by first finding the two recall / precision pairs (r_1, p_1) and (r_2, p_2) that bracket the breakeven point. We then use linear interpolation to find where on the line defined by the bracketing points recall and precision would be equal:

$$r = p = \frac{r_2 \times p_1 - r_1 \times p_2}{r_2 - r_1 + p_1 - p_2}$$

For example, the breakeven point for the microaveraged values in Table 8.1 is

$$r = p = \frac{.30 \times .30 - .28 \times .29}{.30 - .28 + .30 - .29}$$

or .29, rounded to two decimal places. Linear extrapolation from the two nearest points was used when bracketing points were not available.

Returning to the issue of strategies for producing a binary categorization from probability estimates, we investigated two methods in addition to probability thresholding. For these experiments, we only used the feature set that excluded stopwords, due to its clear superiority. One strategy, *k-per-document*, was to assign a fixed number of categories per test document, as Maron did. Unlike Maron, we did not make use of knowledge of the correct number of categories to assign to each document. We simply assigned the highest scoring category, the two highest scoring categories, and so on to each document. Precision was traded for recall by increasing the number of categories assigned to each document. The results are shown in Table 8.3.

The third strategy tried was *proportional assignment*. The idea was to try a strategy analogous to k-per-document, but comparing documents to each other within categories rather than categories within documents. Since categories vary so widely in frequency, assigning every category the same number of times clearly did not make any sense. Instead, we chose to assign each category in proportion to the number of times it was assigned on the training corpus, and then to vary the proportionality constant to trade off precision against recall. Results are as shown in Table 8.4.

We then experimented with Maron's strategy of scaling the probability estimates for each document to sum to 1.0 or, rather, the approximation to this strategy we described in the previous section. It is interesting to note that the presence or absence of scaling would not have affected Maron's own results, since the ordering of category scores for any particular document is unchanged. Similarly, it does not affect our results for the k-per-document strategy. It does however affect our results for probability thresholding and proportional assignment, and we present these results in Tables 8.5 and 8.6.

Table 8.1 Effectiveness for probability thresholding categorization strategy. Column 2 shows the number of assignments that should be made to 723 test documents to achieve the same ratio of category assignments to documents as on the training corpus. Column 3 shows the probability threshold that produces the desired number of assignments. No stop list was used.

	Desired No.	Log Prob	Microaveraged		Macroaveraged	
Proportion	Assigned	Threshold	Recall	Precision	Recall	Precision
0.05	23	22.59000	.01	.52	.01	.39
0.10	46	20.10000	.03	.59	.03	.50
0.20	93	16.70000	.06	.57	.06	.54
0.40	185	13.02600	.10	.49	.09	.45
0.50	232	12.19000	.12	.47	.11	.43
0.60	278	11.56873	.14	.45	.12	.42
0.80	371	10.37000	.17	.41	.13	.38
1.00	464	9.23000	.20	.39	.14	.35
1.20	556	8.30290	.22	.36	.15	.34
1.40	649	7.68000	.24	.33	.16	.30
1.60	742	6.97230	.26	.32	.18	.29
1.80	835	6.44600	.28	.30	.19	.30
2.00	927	6.01600	.30	.29	.20	.28
2.50	1159	5.09000	.34	.26	.22	.25
3.00	1391	4.30700	.37	.24	.24	.23
4.00	1855	2.97400	.45	.22	.28	.24
5.00	2318	2.01700	.50	.19	.31	.20
10.00	4637	−1.37310	.70	.13	.44	.14
15.00	6955	−3.51120	.78	.10	.50	.10
20.00	9273	−5.25140	.82	.08	.53	.08
25.00	11592	−6.94640	.86	.07	.57	.07
		Breakeven:	.29		.24	

Table 8.2 Effectiveness for probability thresholding categorization strategy. Column 2 shows the number of assignments that should be made to 723 test documents to achieve the same ratio of category assignments to documents as on the training corpus. Column 3 shows the probability threshold that produces the desired number of assignments. A stop list was used.

	Desired No.	Log Prob	Microaveraged		Macroaveraged	
Proportion	Assigned	Threshold	Recall	Precision	Recall	Precision
0.05	23	22.77000	.01	.52	.01	.52
0.10	46	19.22000	.03	.50	.02	.52
0.20	93	16.58000	.06	.56	.05	.56
0.40	185	13.60400	.12	.59	.10	.61
0.50	232	12.08000	.15	.58	.12	.63
0.60	278	11.20000	.18	.58	.13	.61
0.80	371	9.71000	.22	.52	.15	.50
1.00	464	8.39400	.25	.49	.17	.40
1.20	556	7.36600	.28	.45	.18	.40
1.40	649	6.58400	.31	.43	.21	.36
1.60	742	5.96000	.35	.42	.22	.37
1.80	835	5.41000	.36	.39	.24	.35
2.00	927	5.00300	.39	.37	.25	.35
2.50	1159	4.01000	.43	.34	.27	.33
3.00	1391	3.26400	.48	.31	.30	.30
4.00	1855	2.00000	.55	.26	.34	.25
5.00	2318	0.97500	.60	.23	.36	.21
10.00	4637	−2.18000	.74	.14	.47	.13
15.00	6955	−4.33720	.81	.11	.54	.10
20.00	9273	−6.22100	.85	.08	.58	.08
25.00	11592	−7.95850	.88	.07	.61	.06
		Breakeven:	.38		.30	

Table 8.3 Results from assigning a fixed number of categories to each document (k-per-doc). A stop list was used.

	Microaveraged		Macroaveraged	
No. Per Doc	Recall	Precision	Recall	Precision
1	.40	.49	.24	.42
2	.54	.34	.32	.30
3	.64	.26	.39	.24
4	.68	.21	.41	.20
5	.72	.18	.44	.18
6	.75	.15	.46	.16
7	.77	.14	.48	.15
8	.79	.12	.50	.13
9	.81	.11	.51	.11
10	.82	.10	.52	.10
11	.83	.09	.53	.09
12	.84	.09	.54	.09
13	.86	.08	.56	.08
14	.86	.08	.57	.08
15	.87	.07	.57	.08
16	.88	.07	.58	.07
Breakeven:	.44		.31	

8.5 Analysis

The effectiveness of all strategies was surprisingly poor. We had expected that the large amount of training data available would make quite accurate the relative magnitudes of the probability estimates, if not their exact values, and so expected that probability thresholding would work quite well.

Several possibilities for the poor effectiveness occurred to us. One was the simplistic feature selection method used. We had attempted to replicate Maron's strategy to the extent possible, without human intervention, but it seems likely that Maron's own judgment in picking features was important. Whether or not we accurately replicated what Maron's strategy had been with respect to peaks, this strategy eliminated very few features from consideration on the Reuters collection, since the large number of documents allowed peaks to occur accidentally. The result was that essentially the most frequent 286 words, or the most frequent 286 non-stopwords, were used as features. This included many low content words.

Another possible reason for the poor effectiveness of probability thresholding was the incomparability of probability estimates across documents. Unlike Maron's collection of abstracts, the Reuters documents vary widely in length, so the longer documents are likely to get much higher scores for all categories than shorter documents. The k-per-document method was inspired by this observation, and achieved

Table 8.4 Results from assigning categories in same proportion on training and test set, ranking documents within each category (proportional assignment). A stop list was used.

	Microaveraged		Macroaveraged	
No. Per Doc	Recall	Precision	Recall	Precision
0.05	.01	1.00	.01	1.00
0.10	.03	.96	.01	.86
0.20	.06	.75	.02	.67
0.40	.10	.57	.03	.50
0.50	.11	.52	.04	.42
0.60	.14	.49	.06	.40
0.80	.17	.46	.07	.38
1.00	.21	.44	.08	.37
1.20	.24	.41	.10	.38
1.40	.26	.38	.11	.36
1.60	.28	.36	.13	.36
1.80	.30	.33	.15	.35
2.00	.32	.32	.16	.35
2.50	.35	.28	.18	.31
3.00	.39	.26	.20	.28
4.00	.45	.22	.23	.23
5.00	.50	.20	.28	.21
10.00	.63	.15	.36	.11
15.00	.70	.12	.42	.08
20.00	.75	.11	.45	.06
25.00	.78	.09	.48	.05
Breakeven:	.32		.23	

Table 8.5 Probability thresholding, with scaling of probability estimates to sum to 1.0 across each document. A stop list was used.

	Microaveraged		Macroaveraged	
No. Per Doc	Recall	Precision	Recall	Precision
0.05	.02	.65	.02	.59
0.10	.03	.65	.03	.62
0.20	.08	.76	.08	.76
0.40	.15	.75	.12	.73
0.50	.19	.72	.14	.66
0.60	.21	.69	.16	.63
0.80	.27	.64	.19	.60
1.00	.31	.61	.21	.58
1.20	.36	.58	.23	.55
1.40	.39	.53	.24	.48
1.60	.40	.48	.24	.42
1.80	.42	.45	.26	.40
2.00	.45	.43	.27	.37
2.50	.49	.38	.30	.33
3.00	.53	.34	.32	.28
4.00	.59	.29	.36	.24
5.00	.63	.24	.39	.21
10.00	.74	.14	.49	.12
15.00	.80	.10	.53	.08
20.00	.83	.08	.56	.06
25.00	.86	.07	.61	.05
Breakeven:	.44		.31	

Table 8.6 Results from assigning categories in same proportion on training and test set, ranking documents within each category (proportional assignment). Scaling of probability estimates to sum to 1.0 across each document is used. A stop list was used.

	Microaveraged		Macroaveraged	
No. Per Doc	Recall	Precision	Recall	Precision
0.05	.01	1.00	.01	1.00
0.10	.03	.93	.01	.71
0.20	.06	.74	.02	.55
0.40	.10	.61	.04	.64
0.50	.13	.57	.05	.59
0.60	.15	.54	.06	.51
0.80	.19	.51	.08	.50
1.00	.22	.48	.10	.47
1.20	.26	.45	.12	.44
1.40	.28	.40	.13	.42
1.60	.30	.39	.15	.44
1.80	.33	.37	.17	.41
2.00	.35	.35	.19	.41
2.50	.37	.30	.21	.35
3.00	.42	.28	.23	.32
4.00	.48	.24	.26	.25
5.00	.53	.21	.30	.22
10.00	.66	.15	.40	.12
15.00	.71	.13	.45	.08
20.00	.75	.11	.46	.06
25.00	.78	.09	.49	.05
Breakeven:	.35		.26	

Table 8.7 Proportion of documents with TOPIC categories assigned: Carnegie Group's training vs. test set division.

	No. Docs	No. Docs w/ 1+ TOPIC	% w/ 1+ TOPIC	No. TOPICS Assigned	Ave. TOPICS Per Doc
Training	21,450	11,098	51.7%	13,756	0.64
Test	723	566	78.3%	896	1.24

effectiveness considerably better than that of probability thresholding (Table 8.2 vs. Table 8.3).

Another approach to dealing with the incomparability of scores between documents was to introduce scaling of scores for each document. This improved the effectiveness of probability thresholding a moderate amount (Table 8.2 vs. Table 8.5).

Alternately, we could attribute the problems with probability thresholding to incomparability of scores across categories. This possibility inspired the proportional assignment strategy (Table 8.4), which achieved effectiveness somewhat worse than that of probability thresholding (Table 8.2), and also improved less with scaling of scores (Table 8.6).

Despite its unexciting level of effectiveness, our experiment with the proportional assignment method led to some valuable insights about the Reuters collection. Proportional assignment was tried by hand first, as we had not originally written code for it. This began by first finding (using transaction files and Unix utilities) the frequency of each category on the training set. Computing these values revealed that only 116 of the 118 selected categories had transactions on the training set. In other words, there were two categories for which no training data was available. This is a realistic situation for categorization systems, since categories may be defined before any texts appropriate to the category have been seen. No purely learning-based system can assign such categories.

In any case, this observation suggested that it was rather arbitrary to include the two categories that had test instances but no training instances, while omitting categories that occurred in neither test nor training documents. So one decision we made was to use the full set of 135 categories in the future.

Returning to the issue of proportional assignment, we needed to know the proportion of all assignments for which a particular category accounted. The maximum likelihood estimate for this value is the number of assignments for category C_j on the training set, divided by the number of assignments of all categories on the training set. The total number of assignments of all TOPIC categories on the training set was 13,576, which drove home that many of the training documents have no TOPIC assignments at all.[5] This made us wonder about the test documents, so we analyzed the transaction files with the result shown in Table 8.7.

[5] All but a few documents have some category assigned, often a COUNTRY category if nothing else.

Table 8.8 Proportion of category assignments accounted for by 10 most frequent categories from training set plus 10 most frequent from test set (Carnegie Group test/training partitioning).

Category and ID	Training Set (21,450 docs)		Test Set (723 docs)	
	No. Assigned	% of 13,756 Assignments	No. Assigned	% of 896 Assignments
EARN	4,053	29.5%	22	2.5%
DLR	185	1.3%	42	4.7%
MONEY-FX	745	5.4%	82	9.2%
INTEREST	497	3.6%	33	3.7%
GNP	141	1.0%	28	3.1%
ACQ	2,427	17.6%	79	8.8%
SHIP	290	2.1%	18	2.0%
WHEAT	293	2.1%	23	2.6%
GRAIN	601	4.4%	45	5.0%
CORN	240	1.7%	15	1.7%
CRUDE	585	4.3%	54	6.0%
NAT-GAS	108	0.8%	25	2.8%
TRADE	504	3.7%	65	7.3%
Total	10,669	77.6%	531	59.3%

We can see that a significantly higher proportion of test documents had TOPIC categories than did training documents, and that almost twice as many TOPIC categories were assigned to test documents than training documents on the average. This made us wonder if the distribution as well as the number of categories on the test set might be different. We examined the number of occurrences of the 10 most frequent categories assigned to the training set and the 10 most frequent on the test set, resulting in a total of 13 categories, as shown in Table 8.8.

The differences are quite striking. The two highest frequency categories are systematically underrepresented in favor of other categories. Hayes (personal communication) believes that these differences result from fluctuations over time in the kinds of stories that appear on the newswire. For instance, earnings reports on companies occur at specific times of the year. The test stories were drawn from texts appearing on a relatively small number of days, so that the test set formation was very susceptible to such fluctuations.

The fact that proportional assignment did reasonably well despite the atypical category distribution on the test set, suggests it is an appropriate strategy to use, and should perform better with a more typical test set. It should also perform better as the test set becomes larger, since the distributional properties of such a test set will more closely approximate the properties of the population as a whole.

Preparation of data for proportional categorization required that for each category we sort the set of test documents according to their score on that category. When two documents had the same score, it was necessary to order them arbitrarily.

The ranking program outputs a warning indicating how many documents had tied scores. The result was that every category had exactly 46 tied scores. At first this seemed like an incredible coincidence (i.e. a bug). However, we realized that if two documents had the same set of predictor feature values then, since we used the same predictor features for all categories, they would necessarily have the same score on every category. While two documents could in theory have tied scores without sharing all feature values, in practice the use of 32-bit scores and high frequency features made this unlikely.

Out of curiosity, we examined the documents with tied scores and discovered that they came in 23 pairs. The members of each pair turned out to be stories that were almost identical. One was a minor variation on the other, with perhaps one sentence or one word changed, or even no apparent changes. This was a worrisome development from the standpoint of evaluation. If, out of 723 documents, there were 23 pairs of documents with identical presence or absence of 286 different words, there were might be many more pairs or larger groups of near duplicates that varied on just a few of those 286 words. Given that the coverage of all but the highest frequency categories on the test set was quite low, the possibility that there were effectively fewer than 723 test documents was worrisome.

Furthermore, if there was this much near duplication in the test set, there might also be considerable near duplication between the training set and the test set. We could be training on our test data to a significant degree without knowing it. We therefore decided that a new test set would have to be a priority before further experimentation.

8.6 Insights From Another Data Set

Subsequent to the above-mentioned experiments, we performed another set of text categorization experiments in the context of the MUC-3 evaluation of natural language processing systems [Lew91a]. The experiments involved assigning 100 test newswire stories and other narrative texts to 88 content-based categories. The proportional assignment strategy described above was used with a Bayesian classifier similar though not identical to the one above. However, feature selection was radically changed. A mutual information measure was used to select a distinct set of predictor features for each category. Effectiveness on the order of 50% recall and precision (microaveraged) was achieved despite a training set of only 1400 documents. We therefore decided to use a mutual-information-based feature selection strategy in the remaining Reuters experiments, as described in the following chapters.

Another result from our MUC-3 experiments was that effectiveness varied relatively little across a wide range of feature set sizes. This suggested that the large feature sets used in the above experiments might not be needed, particularly with a better feature selection mechanism.

The MUC-3 evaluation also contributed to our worries about the problem of near duplicate documents. The repetition of descriptions of particular events in the MUC-3 corpus was quite common, including near duplicates of the kind found in the

Reuters corpus. This led to phenomena such as the word *Jesuits* being picked by mutual information as a good predictor feature for the category *MURDER*, because of the large number of training documents reporting on a particular incident of Jesuit priests being murdered. Questions were raised in discussions at the MUC-3 workshop as to whether machine-learning-based systems were benefiting from near duplication of text from the training set in the test set, something that might not be as prevalent in an operational setting. This contributed to our decision to reconsider the training/test set division for the Reuters collection.

8.7 Summary

The experiments reported in this chapter were intended to uncover any difficulties in the categorization process that might complicate using text categorization to compare text representations. The replication of Maron's classic experiment was chosen as the vehicle for this investigation. Overall effectiveness was disappointing, given the large number of training instances available.

The main contributor to poor effectiveness was a problem that had been finessed in Maron's original experiment: how to use probability estimates to assign categories when each document belongs to an unknown number of possible categories, or to no category at all. This problem has been ignored in most research on statistical text categorization, and based on the above results we believe that this should be a central problem for future research.

Contributing to the difficulty of producing effective binary categorizations were the wildly inaccurate probability estimates produced by the Bayesian formula. While the lack of well-established methods for incorporating dependence information into probabilistic text retrieval models suggests that the problem is difficult enough that we should not take it on for this study, it seemed worthwhile to see whether using a more recent independence model might improve effectiveness. This is attempted in the next chapter.

Another contributor to poor effectiveness was the weak feature selection mechanism used. This was an easy target for improvement, since a number of automated feature selection methods are known in machine learning and information retrieval.

Perhaps the most significant finding of this initial set of experiments was the discovery that distribution of categories was significantly different between the training and test corpora used in the Carnegie Group experiments. This fact, in addition to worries about near duplicate documents occurring on the training and test sets, led us to conclude that a new training/test set division was needed for our remaining experiments.

Given the large number of questions about the text categorization process raised by the results of this chapter, and the changes they suggested in our methods, we decided that a further round of experiments on categorization was necessary before returning to our hypotheses about text representation. These further experiments are reported in the next chapter.

a	but	himself	not	t	whensoever
about	by	his	nothing	than	where
above	c	how	now	that	whereafter
across	can	however	nowhere	the	whereas
after	cannot	i	o	their	whereat
afterwards	co	ie	of	them	whereby
again	could	if	off	themselves	wherefrom
against	d	in	often	then	wherein
albeit	down	inc	on	thence	whereinto
all	during	indeed	once	there	whereof
almost	e	into	one	thereafter	whereon
alone	each	is	only	thereby	whereto
along	eg	it	onto	therefore	whereunto
already	either	its	or	therein	whereupon
also	else	itself	other	thereupon	wherever
although	elsewhere	j	others	these	wherewith
always	enough	k	otherwise	they	whether
among	etc	l	our	this	whither
amongst	even	last	ours	those	which
an	ever	latter	ourselves	though	whichever
and	every	latterly	out	through	whichsoever
another	everyone	least	over	throughout	while
any	everything	less	own	thru	whilst
anyhow	everywhere	ltd	p	thus	who

Figure 8.2 The stopword list used (part 1).

anyone	except	m	per	to	whoever
anything	f	many	perhaps	together	whosoever
anywhere	few	may	q	too	whomever
are	first	me	r	toward	whomsoever
around	for	meanwhile	rather	towards	whole
as	former	might	s	u	whom
at	formerly	more	same	under	whose
b	from	moreover	seem	until	why
be	further	most	seemed	up	will
became	g	mostly	seeming	upon	with
because	h	much	seems	us	within
become	had	must	several	v	without
becomes	has	my	she	very	would
becoming	have	myself	should	via	v
been	he	n	since	w	w
before	hence	namely	so	was	x
beforehand	her	neither	some	we	yet
behind	here	never	somehow	well	you
being	hereafter	nevertheless	someone	were	your
below	hereby	next	something	what	yours
beside	herein	no	sometime	whatever	yourself
besides	hereupon	nobody	sometimes	whatsoever	yourselves
between	hers	none	somewhere	when	z
beyond	herself	noone	still	whence	
both	him	nor	such	whenever	

Figure 8.3 The stopword list used (part 2).

CHAPTER 9

TEXT CATEGORIZATION: SECOND EXPERIMENTS

The experiments reported in the previous chapter cast serious doubt on the use of the original training / test set division for the Reuters corpus. They also suggested a number of ways in which our categorization algorithms could be improved. In this chapter, we describe experiments with a new training / test set division, while also exploring a number of potential improvements in the categorization algorithms. The new categorization procedure achieves considerably better effectiveness than the one reported in the previous chapter, and appears to be an appropriate vehicle for investigating representation quality.

9.1 A New Training / Test Partitioning

In the previous chapter we mentioned three major problems with the original training / test set division on the Reuters corpus: the potential for near duplication between training and test stories, the small size of test set, and the very different distribution of category assignments on the training and test set.

Of these three problems, near duplication imposed the tighest constraints on a new training / test division. All the examples of near duplication we have found so far have been stories that appeared on the newswire on the same day, so a partitioning that kept all stories from the same day in the same group seemed the best approach to eliminate most or all cases of near duplication.

Beyond the question of documents that are minor variants of a single original article, there was the larger question of how well the generalizations that are being made by the system could be expected to hold up over time. Our MUC-3 results, for instance, provided examples of inappropriate generalizations resulting from a single real world event repeatedly mentioned in newswire stories over a period of time. It is not clear how best to take the time-varying nature of a text stream into account when evaluating a text categorization system.

In the long run, the need to cope with such issues suggests that incremental machine learning algorithms will play an important role in content-based text processing. For this project, which was concerned with evaluating batch mode learning methods, the safest approach was to adopt a training / test division obtainable in an operational setting, i.e. one where all documents in the training set appear chronologically before those in the test set.

Table 9.1 Proportion of category assignments accounted for by 10 most frequent categories from training set plus 10 most frequent from test set (new training / test partition).

Category and ID	Training Set (14,704 docs)		Test Set (6,746 docs)	
	No. Assigned	% of 9,631 Assignments	No. Assigned	% of 4,125 Assignments
EARN	2,877	29.87%	1,176	28.51%
MONEY-FX	538	5.59%	207	5.02%
INTEREST	347	3.60%	150	3.64%
ACQ	1,651	17.14%	776	18.81%
SHIP	198	2.06%	92	2.23%
WHEAT	212	2.20%	81	1.96%
GRAIN	433	4.50%	168	4.07%
CORN	176	1.83%	64	1.55%
CRUDE	388	4.03%	197	4.78%
TRADE	369	3.83%	135	3.27%
Total	7,189	74.64%	3,046	73.84%

Once a chronological division was assumed, the only remaining issue was where in the chronological sequence of stories the division should be made. Ideally one would like a substantial number of both training and test examples for every category. Unfortunately, in the Reuters corpus, as in many real-world categorization situations, many of the categories had few or no examples. The best we could do was to get a reasonable number of training and test examples for the more frequent categories. We chose a roughly two-thirds / one-third split as producing a substantial number of categories with 20 or more of both training and test examples. The final division was such that all stories from April 7, 1987 and earlier went into a set of 14,704 training stories, and all stories from April 8, 1987 or later went into a test set of 6,746 stories. Further details on this partitioning can be found in Table 9.2.

The original 723 test stories were omitted completely from both the training and test set, because of their role in past tests of the CONSTRUE system. At some future date we plan to compare the effectiveness of statistical categorization algorithms with the original CONSTRUE effectiveness data. The credibility of such a comparison will be increased if our investigations into categorization algorithms are not guided by effectiveness on the original CONSTRUE test set.

Having chosen the new partition to satisfy the demands of chronological ordering and a large test set, we needed to check whether the third problem, that of category distribution had also been solved. As Table 9.1 shows, the distributional properties are indeed very similar for the 10 most frequent categories on the new training and test set. This is in marked contrast to the situation with the Carnegie Group training / test set division, and it appears that any systematic bias has been avoided.

9.2 Categorization Techniques

In this section we discuss two improvements to our categorization procedure motivated by the results in the previous chapter. The most important of these is the use of a statistical feature selection algorithm to choose a separate set of predictor words for each category. A secondary improvement was the making of several changes to our Bayesian classification formula.

9.2.1 Feature Selection

As we have mentioned a number of times, the issue of feature selection is an important one for text classification systems. In text retrieval systems, the user request is used to choose features. For text categorization systems there is no request, so the system builder must select a set of features, either manually or by using some automatic method.

In replicating Maron's study in the previous chapter, we attempted to use the same constraints on features that he used, but did not attempt to duplicate his human judgment. However, better mechanical feature selection methods are available than were used in our replication. Maron himself, as well as Hamill and Zamora [HZ80] suggested, but did not try, information-theoretic measures for selecting predictor words. Such measures are widely used for feature selection in pattern recognition and machine learning [Kit86]. We chose to use $I(W_i; C_j)$, the *system mutual information* [Ham80] between assignment or nonassignment of category C_j and assignment or nonassignment of feature W_i, as a feature quality measure:

$$I(W_i; C_j) = \sum_{b=0,1} \sum_{c=0,1} P(W_i = b, C_j = c) log_2 \frac{P(W_i = b, C_j = c)}{P(W_i = b) \times P(C_j = c)} \qquad (9.1)$$

This measure has also been refered to as the *expected mutual information measure (EMIM)* [van77]. Notice that system mutual information is symmetric with respect to C_j and W_i. However, the above formula is sometimes rewritten to emphasize the information gained about the feature we desire to predict (C_j in our case) based on the feature we can observe (W_i):

$$\left(\sum_{c=0,1} F(P(C_j = c)) \right) - \left(\sum_{b=0,1} P(W_i = b) \sum_{c=0,1} F(P(C_j = c | W_i = b)) \right) \qquad (9.2)$$

where $F(p) = p \log_2 p$. In the latter form the measure has been referred to as *information gain* [Qui86a].

For each category, the words with the top d values on system mutual information were selected as features. This had the important effect that, unlike in the experiments in the previous chapter, a different set of features was used in predicting each category.

A secondary issue was how many features should be chosen for each category. Our experiments with the MUC-3 corpus showed that effectiveness was quite similar

across a wide range of feature set sizes, including ones lying outside Fuhr's suggested range of 1 feature per 50 to 100 training instances. Section 9.3 therefore includes experiments on a variety of feature set sizes.

9.2.2 Classification Formula

In the previous chapter we used Maron's formula for categorization essentially as given. Here we look more closely at the theoretical underpinnings of the formula, and address a recent claim by Cooper about the logical inconsistency of these underpinnings. We then present an alternative approach in which the presence or absence of predictor words is not assumed to be known with certainty, and briefly consider several issues in probability estimation.

9.2.2.1 Theoretical Background

In this section we look at the methods for probability estimation underlying Maron's formula. The principle that Maron's estimation formula was based on is Bayes' theorem ([Jam85], p. 11):

$$P(C_j = a_{ji}|\mathbf{x}) = \frac{P(\mathbf{x}|C_j = a_{ji})P(C_j = a_{ji})}{\sum_k P(\mathbf{x}|C_j = a_{jk})P(C_j = a_{jk})} \tag{9.3}$$

The summation in the denominator is over all values a_{jk} that the random variable C_j can take on. For text categorization we can specialize this to the case where C_j is binary. Rewriting and simplifying we have:

$$P(C_j = 1|\mathbf{x}) = \frac{P(\mathbf{x}|C_j = 1)P(C_j = 1)}{P(\mathbf{x})} \tag{9.4}$$

In the case of Maron's formula for document categorization, the conditioning event $\mathbf{x}$ is the presence and absence of predictor words in a document to be categorized. If $W_1, W_2, \ldots, W_p$ are a set of p words chosen as predictor features, then we have an event space of 2^p possible conditioning events, since each of the p words can be either present or absent. Making explicit that $\mathbf{x}$ is a set of outcomes on $W_1, W_2, \ldots, W_p$, we can rewrite Formula 9.4 as:

$$P(C_j = 1|W_1 = x_1, \ldots, W_p = x_p) = \frac{P(W_1 = x_1, \ldots, W_p = x_p|C_j = 1)P(C_j = 1)}{P(W_1 = x_1, \ldots, W_p = x_p)} \tag{9.5}$$

In this form, the formula is not particularly useful for text categorization. We are unlikely to see enough training documents with exactly the set of occurrences $W_1 = x_1, \ldots, W_p = x_p$ to estimate the probabilities $P(W_1 = x_1, \ldots, W_p = x_p|C_j = 1)$ and $P(W_1 = x_1, \ldots, W_p = x_p)$ with any accuracy. What we would like is a method of allowing the probabilities associated with each predictor word to be estimated individually, and then combined into an overall estimate. The usual approach to this has been to make assumptions about the independence of the probabilities being

estimated. Maron explicitly makes the assumption that the occurrence of any pair of words is conditionally independent given that a document belongs to a particular category:

$$P(W_i = 1, \ldots, W_k = 1 | C_j = 1) = P(W_i = 1 | C_j = 1) \times P(W_k = 1 | C_j = 1) \quad (9.6)$$

for all W_i, W_k, C_j. This implies:

$$P(W_1 = x_1, \ldots, W_p = x_p | C_j = 1) = P(W_1 = x_1 | C_j = 1) \times \ldots \times P(W_1 = x_1 | C_j = 1) \quad (9.7)$$

for all $x_1, \ldots, x_p$, given that $W_1, \ldots, W_p$ are binary. This enables us to estimate the numerator in Formula 9.5 from a reasonable amount of data. Maron avoided the issue of estimating the denominator in Formula 9.5 by using the same set of predictor words for each category and by using only the ordering of probability estimates for categories, rather than their actual values. In order to use Formula 9.5 to produce actual probability estimates, rather than just rankings, when the amount of data is limited, an additional assumption is needed. The usual assumption made in IR is that the occurrence of any pair of words is independent absolutely:

$$P(W_1 = x_1, \ldots, W_p = x_p) = P(W_1 = x_1) \times \ldots \times P(W_p = x_p) \quad (9.8)$$

Taking assumptions 9.6 and 9.8 together means we can replace Formula 9.5 with:

$$P(C_j = 1 | W_1 = x_1, \ldots, W_p = x_p) = P(C_j = 1) \times \prod_i \frac{P(W_i = x_i | C_j = 1)}{P(W_i = x_i)} \quad (9.9)$$

Maron made the further simplification of only treating the presence ($W_i = 1$) and not the absence ($W_i = 0$) of clue words as being significant, replacing Formula 9.9 with:

$$P(C_j = 1 | d) = P(C_j = 1) \times \prod_{i_d} \frac{P(W_i = 1 | C_j = 1)}{P(W_i = i)} \quad (9.10)$$

where for document d, i_d ranges only over those values such that $W_{i_d} = 1$. This approach treats the lack of a term in a document as equivalent to having no information about whether the term takes on the value 0 or 1 for the document. This is the formula used in the experiments in the previous chapter.

9.2.2.2 Validity of Assumptions

In a recent article [Coo91], Cooper stresses the point that assumption 9.8 is not needed in a probabilistic retrieval system if only ranking is desired. He further points out that assumptions 9.6 and 9.8 can both be true only for a very narrow range of probability distributions, which do not in general fit the empirical data found in IR systems. He presents an example where making assumptions 9.6 and 9.8 in the context of a hypothetical empirical probability distribution results in probability estimates greater than 1.0.

Cooper then claims that this is a logical inconsistency, and should destroy the usefulness of the method. He also claims that the only explanation for the previous

success of probabilistic retrieval systems is that, since they use only rankings, they really rely only on assumption 9.6.

The problem is actually both more and less severe than Cooper claims. It is more severe in that there is little reason to believe that even assumption 9.6 holds for empirical data encountered in IR. To take an example from our text categorization domain, we have 14704 training documents, of which 1651 are assigned the ACQ (corporate acquisitions category). Of those 1651 documents, 52 contain the word *joint*, 53 contain the word *venture*, and 32 contain both words. Using a maximum likelihood estimate for all probabilities, we would estimate:

$$\mathrm{P}(\ \textit{joint} = 1 \mid \mathrm{ACQ} = 1\ ) = \frac{52/14704}{1651/14704} = 0.031$$
$$\mathrm{P}(\ \textit{venture} = 1 \mid \mathrm{ACQ} = 1\ ) = \frac{53/14704}{1651/14704} = 0.032$$
$$\mathrm{P}(\ \textit{joint} = 1, \textit{venture} = 1 \mid \mathrm{ACQ} = 1\ ) = \frac{32/14704}{1651/14704} = 0.019$$

But under assumption 9.6 we should have:

$$\begin{aligned}&\mathrm{P}(\ \textit{joint} = 1, \textit{venture} = 1 \mid \mathrm{ACQ} = 1\ )\\ &\quad = \mathrm{P}(\ \textit{joint} = 1 \mid \mathrm{ACQ} = 1\ ) \times \mathrm{P}(\ \textit{venture} = 1 \mid \mathrm{ACQ} = 1\ )\\ &\quad = 0.031 \times 0.032\\ &\quad = 0.001 << 0.019\end{aligned}$$

Many examples like this could be provided. The probability estimates that result do not exceed 1.0, as they can be made to do when both assumptions 9.6 and 9.8 are allowed, but is just as clear that the single assumption 9.6 is not consistent with the empirical data.

On the other hand, the situation is also less drastic than Cooper would have us believe. We use probabilitistic theories of information retrieval not to prove statements about IR systems, but to produce approximations to an optimal ranking or categorization of documents. The proper question is not whether the exact, and probably unknowable, probability distribution has been found, but how good an approximation to that distribution we have.

We believe advances in probabilitistic retrieval and categorization are most likely to come from attempts to deal directly with the dependencies that clearly are present in our data, through techniques such as term clustering, phrasal indexing, and retrieval models incorporating dependencies. In this study we are investigating the use of the first two methods.

9.2.2.3 Probabilistic Indexing

In the preceding discussion we assumed, as in Maron's paper, that we know the event $W_1 = x_1, \ldots, W_p = x_p$ to condition on in $P(C_j = 1 | W_1 = x_1, \ldots, W_p = x_p)$. In other words, we assume that, for a particular document, we know with certainty what values our predictor features $W_1, \ldots, W_p$ take on. If we define these features to correspond, for instance, to the presence or absence of particular sequences of

ASCII characters in the part of the document that we have available online, then this might be reasonable.

On the other hand suppose, as is more commonly the case, we want the feature W_i to indicate that a particular word, say *banking*, is strongly related to the ideas conveyed by a document. Certainly the presence of the string "*banking*" in a document is an excellent indication that the word *banking* is related to the content of that document. But several occurrences of that string might be an even stronger indication. One occurrence would be stronger, perhaps, if the document were shorter. The presence of the string "*bznking*" would also be good evidence, particularly if the document had been entered by optical character recognition. Even if the document does not appear to contain any string that plausibly was intended to be "*banking*", the author, if given more space to write about the same topic, might have used the word *banking* at some point.

The idea that we may be uncertain about whether an index term should be assigned to a document, and that we may want to take this uncertainty into account when retrieving or categorizing documents, is called *probabilistic indexing*. Probabilistic indexing, and the more general idea of weighted assignment of indexing terms, has been widely investigated in information retrieval [MK60, BS74, SM83, Cro83, Fuh89, Tur90].

Bayesian classification can be adapted to support probabilistic indexing. One approach is presented by Fuhr [Fuh89]. Fuhr distinguishes between a document D_m, and a *correct indexing*, $\mathbf{x}$, of D_m. The notion of correctness is only loosely defined, but the idea is that given a set of binary indexing terms W_i, there is some ideal set of values $W_1 = x_1, \ldots, W_p = x_p$ that should be used to represent the document. We assume, however, that these ideal values are not known, and that we are in position of needing to estimate the probability $P(\mathbf{x}|D_m)$ that a particular vector $\mathbf{x}$ is the ideal indexing of the document. Note that a feature W_i is still considered to only take on the values 0 and 1. We now, however, take into account our uncertainty about which of those two values the feature should have for a particular document.

Fuhr discusses probabilistic indexing in the context of text retrieval. In the following we straightforwardly adapt the approach, with some change of notation, to text categorization. Rather than estimating $P(C_j = 1|\mathbf{x})$ for a known set of feature values $\mathbf{x}$, as we did in the preceding chapter, we now want to estimate $P(C_j = 1|D_m)$ for document D_m. We still characterize D_m in terms of $\mathbf{x}$, its values

on the W_i, but now we assume we only have a probability distribution for what those values might be, and must sum over that discrete probability distribution:

$$P(C_j = 1|D_m) = \sum_{\mathbf{x}\in\{0,1\}^p} P(C_j = 1|\mathbf{x}) \times P(\mathbf{x}|D_m) \tag{9.11}$$

Applying Bayes' Theorem yields:

$$P(C_j = 1|D_m) = \sum_{\mathbf{x}\in\{0,1\}^p} P(C_j = 1) \times \frac{P(\mathbf{x}|C_j = 1)}{P(\mathbf{x})} \times P(\mathbf{x}|D_m) \tag{9.12}$$

If we once again make assumptions 9.6 and 9.8, and in addition assume

$$P(\mathbf{x}|D_m) = P(x_1|D_m) \times \ldots \times P(x_p|D_m) \tag{9.13}$$

which expanded for clarity is:

$$P(W_1 = x_1, \ldots, W_p = x_p|D_m) = P(W_1 = x_1|D_m) \times \ldots \times P(W_p = x_p|D_m) \tag{9.14}$$

we then get the following formula for $P(C_j = 1|D_m)$:

$$P(C_j = 1) \times \prod_i \left(\frac{P(W_i = 1|C_j = 1) \times P(W_i = 1|D_m)}{P(W_i = 1)} + \frac{P(W_i = 0|C_j = 1) \times P(W_i = 0|D_m)}{P(W_i = 0)} \right) \tag{9.15}$$

This is the formula used in the remainder of the experiments in this dissertation, as well as in our MUC-3 experiments [Lew91a].

9.2.2.4 Estimation

Fuhr's formula requires several probabilities to be estimated, just as Maron's did. In Chapter 8 we followed Maron's model of using the maximum likelihood estimator for the probability of event A:

$$P_{est}(A) = \frac{N(A)}{N(A) + N(\neg A)} \tag{9.16}$$

In other words, the probability of an event is estimated by the number, N(A), of observations that result in the event, divided by the total number of observations. The only exception made was that a small nonzero probability was used to replace values of 0 for $P(W_i = 1|C_j = 1)$.

The problem that Maron encountered with 0 values for $P(W_i = 1|C_j = 1)$ is one example of the more general problem of estimating small probabilities. Alternatives are available that are often superior to the maximum likelihood estimator when the number of samples is small. One of the simplest involves adding 0.5 to the number of observations of each member of the set of disjoint events whose probabilities

are being estimated. This method has often been used in information retrieval [RS76] despite some disadvantages [GC90]. We used this method in the remaining experiments in this study, resulting in the estimates:

$$P_{est}(C_j = 1) = \frac{N(C_j = 1) + 0.5}{N(docs) + 1.0} \tag{9.17}$$

$$P_{est}(W_i = 1) = \frac{N(W_i = 1) + 0.5}{N(docs) + 1.0} \tag{9.18}$$

$$P_{est}(C_j = 1, W_i = 1) = \frac{N(C_j = 1, W_i = 1) + 0.5}{N(docs) + 2.0} \tag{9.19}$$

$$P_{est}(C_j = 1, W_i = 0) = \frac{N(C_j = 1, W_i = 0) + 0.5}{N(docs) + 2.0} \tag{9.20}$$

$N(docs)$ is the total number of training documents, while the other quantities are the numbers of training documents satisfying the specified condition. Note that the value 2.0 in the denominator of the estimates for $P_{est}(C_j = 1, W_i = 1)$ and $P_{est}(C_j = 1, W_i = 0)$ arises because we are partitioning the set of possible outcomes into four events: $(C_j = 0, W_i = 0), (C_j = 0, W_i = 1), (C_j = 1, W_i = 0)$, and $(C_j = 1, W_i = 1)$.

In addition to the probabilities listed above, Fuhr's formula also requires estimates of $P(W_i = 1|D_m)$, the probability that an index term actually takes on a value of 1 for a particular document. The appropriate methods for estimating these values are poorly understood. There is no training sample of documents labeled as to whether $W_i = 1$, or even an operational definition of a correct indexing that would tell us when W_i should equal 1.

A variety of forms of evidence have been used in the past to estimate values for $P(W_i = 1|D_m)$ and for term significance weights in general. These include the class of term (single word vs. phrase, for instance), degree of stemming required to reach a canonical form, proximity or grammatical information, location within document, within document frequency, kinds of linguistic evidence for the term, and inverse document frequency [Fag87, Cro88, Fuh89]. The values that a term has on these characteristics for a particular document has been called its *form of occurrence* in that document [Fuh89]. Machine learning techniques have been used to find a good weight to give to each form of occurrence of a term, both for text categorization [BFL+88] and text retrieval [FB90]. Turtle [Tur90] experimented with a variety of formulas for computing $P(W_i = 1|D_m)$ values purely from frequency information. In the context of his experiments on text retrieval using inference networks he found a good overall formula to be ([Tur90], p. 130):

$$P(W_i = 1|D_m) = c + (1 - c) \times tf_{im} \times idf_i \tag{9.21}$$

Here c is a constant between 0 and 1. The value tf_{im} is the traditional term frequency weight based on within document frequency:

$$tf_{im} = \frac{wdf_{im}}{\text{MAX}_k wdf_{km}} \tag{9.22}$$

The value idf_i is a variant of the conventional inverse document frequency scaled to lie between 0 and 1:

$$idf_i = \frac{\ln(\frac{N(docs)}{N(F_i = 1)})}{\ln(N(docs))} \tag{9.23}$$

We use $N(F_i = 1)$ to indicate the number of documents that have a form of occurrence of term W_i meeting some specified criterion. We discuss this further below.

Formula 9.21 is a variant of the widely used $tf \times idf$ weight [SB87]. A major difference is that this formula assigns a nonzero weight to a term even if there are no occurrences of it in a document. For our experiments we used, at Turtle's suggestion (personal communication), a modification of the formula for tf_{im} so that very long documents would not result in very small tf weights:

$$tf_{im} = \begin{cases} 0.5 + 0.5 \times \frac{\ln(wdf_{im})}{\ln(\text{MAX}_k wdf_{km})} & \text{if } wdf_{im} > 0 \\ 0.0 & \text{if } wdf_{im} = 0 \end{cases} \tag{9.24}$$

In Formula 9.23, we used the notation $N(F_i = 1)$ to indicate the number of documents containing a particular form of occurrence of term W_i. This emphasizes the fact that, once we assume a probabilistic indexing model, it is no longer obvious what the value of $N(W_i = 1)$ is. Rather than estimating this value, we took the same approach as most previous work on probabilistic indexing and let $N(F_i = 1)$ stand in for $N(W_i = 1)$ in Formula 9.23, and also in Formulas 9.18 to 9.20. For the experiments in this chapter, we found $N(F_i = 1)$ as we found $N(W_i = 1)$ when probabilistic indexing was not used, by counting the number of documents that contained a particular string of characters.

9.2.2.5 Event Spaces

In the preceding discussions we have used the notion of the assignment or nonassignment of a category or indexing term to a document. We have repeatedly referred, for instance, to the probability of an assignment or nonassignment of a category or term. It is worth pointing out that these concepts are not unambiguous. In particular, at least two choices are available for how the probability of assignment behaves for a set of categories.

In this chapter we have treated each document as a sample from an event space defined by the cross product of $\{0,1\}^q$ (the set of all possible binary category assignments) with $\{0,1\}^p$ (the set of all possible binary term assignments). In

particular, we have required, for all j, that our a priori estimates of the probability of category assignment have this property:

$$P_{est}(C_j = 1) + P_{est}(C_j = 0) = 1 \tag{9.25}$$

However, this is not the only way to model the assignment of categories to documents. Recall that in the preceding chapter we followed Maron in making the estimate:

$$P_{est}(C_j = 1) = \frac{N(C_j = 1)}{\sum_k N(C_k = 1)} \tag{9.26}$$

where $N(C_k = 1)$ is the number of documents to which C_k is assigned. This means that

$$\sum_k P_{est}(C_k = 1) = 1.0 \tag{9.27}$$

Equations 9.25 and 9.27 in general do not both hold, except in the special case where each document is expected to have exactly one category assigned.

Under the first approach, each sample is a document, and a document takes on a value for each C_j. Since each C_j must take on either the value 0 or the value 1, we want Equation 9.25 to hold for each sample and each j. Under the second approach, each sample is a pair consisting of exactly one document and exactly one category, and Equation 9.27 is what we want.

Both models are reasonable. We have adopted the first, except in our replication of Maron's study, because it has been more widely used in the development of probabilistic IR models. The estimates in Formulas 9.19 and 9.20 are motivated by similar considerations.

9.3 Results

The main goal of this set of experiments, as in the last chapter, was to establish whether the categorization method tested was operating effectively enough that it could be used to compare text representations. In addition, we wanted to find a reasonable set of operating parameters for the method. As in the last chapter, the overall approach was to use a Bayesian formula, in this case Formula 9.15 in combination with the various estimation methods discussed in the previous sections, to estimate the probability of assignment of each category to each test document. Again as in the last chapter, we tested a variety of ways of converting these probability estimates into a set of binary categorizations.

A large number of changes have been made in both the categorization methods tested, and in the data on which they are being tested, since the previous chapter. We summarize these changes in Table 9.2. Note that while we are using only words as predictor features, as in the previous chapter, we are using a slightly different method of segmenting text into words. The new method is described in Section 10.3.2. The change was made to facilitate comparison with the next chapter's results on syntactic indexing phrases.

Table 9.2 Comparison of test collection and categorization method between Chapter 8 and Chapter 9 experiments. Experiments in Chapter 10, as well as this chapter, use the conditions listed for Chapter 9.

	Chapter 8	Chapter 9
Training Documents:		
Number	21,450	14,704
No. days represented	48	35
Timespan covered (1987)	Feb. 26 to Oct. 20	Feb. 26 to Apr. 7
Test Documents:		
Number	723	6,746
No. days represented	11	13
Timespan covered (1987)	Apr. 2 to June 16	Apr. 8 to Oct. 20
Test/training partition	Ad hoc	Chronological
Distribution of categories	Different from training	Similar to training
Categories:		
Total	118	135
Number w/ 1+ assign :		
on training	116	112
on test	77	94
on training & test	75	90
Number w/ 20+ assign :		
on training	56	44
on test	13	31
on training & test	13	31
Categorization Method:		
Classification formula	Bayesian (Maron)	Bayesian (Fuhr)
Indexing	binary	prob. or binary
Feature selection	frequency, stoplist	mutual info, stoplist
Feature set	one set	varies with category

In addition, we experimented with changing the proportionality values at which we measured recall and precision, in order to get good coverage of the range of recall and precision values, particularly those where recall and precision are roughly equal.

Note that most of the changes made in the previous chapter's procedure would be expected to degrade effectiveness. Reducing the number of training documents makes the probability estimates less accurate. Making the training / test split chronologically reduces or eliminates any benefit that might have arisen from near duplicate documents. Adding 17 new categories on which there is little or no training data increases the number of chances for incorrect category assignments.

The one change that was likely to benefit effectiveness was that the new test set had a distribution of categories similar to that of the training set. We also hoped, of course, that the changes in the categorization procedure would increase effectiveness.

In the previous chapter, the only experimental variable was the method used to derive a binary categorization from estimates of category assignment probabilities. In this chapter, two additional experimental variables were important. First, the categorization formula 9.15 can be used with or without probabilistic indexing. (Without probabilistic indexing $P(W_i = 1|D_m)$ is simply always 0 or 1.) Probabilistic indexing has often led to effectiveness improvements in text retrieval systems [Cro83, Fuh89], and we wanted to see what effect it had on text categorization. In Table 9.3 we show the effectiveness of proportional assignment with probabilitistic indexing, using three different values for the constant in Formula 9.21.

The results from Table 9.3, using three variants on probabilistic indexing, can be compared with the first column of Table 9.4, which uses binary indexing. Surprisingly, there was almost no difference in effectiveness among the four conditions.

A second experimental variable was the number of features to be chosen by mutual information for each category. This was motivated by our results on the MUC-3 experiments, which suggested that a smaller number of features per category than the 286 we had been using would be appropriate.

We initially compared feature sets of size 10, 30, and 90, as shown in Table 9.4. These results suggested that relatively small feature sets, on the order of size 10, were as effective as larger ones. These experiments also gave us more information on appropriate proportionality thresholds to use. Tables 9.5 and 9.6 show results for 1, 4, 7, 10, and 15 features per category, and suggest that 10 features is close to optimal for this training set.

We used the proportional assignment method with unscaled estimates in the results just presented even though k-per-doc and probability thresholding worked better than proportional assignment in the experiments reported in the last chapter. The reason was that proportional assignment was the method most likely to have benefited from the repartitioning of the corpus and the use of category-specific sets of predictor features. Indeed, comparing the five methods introduced in the previous chapter on the new partitioning with the new classification formula shows that proportional assignment based on unscaled estimates is now the most effective method under a microaveraged evaluation, and is competitive with the other methods under a macroaveraged evaluation. Table 9.7 compares just the breakeven points for the

Table 9.3 Effectiveness of proportional assignment applied to unscaled category assignment probability estimates. Estimates were generated by Fuhr's classification formula using 10 features per category. Results are presented for three different values (0.2, 0.4, and 0.6) for the default value in Turtle's formula for probabilistic indexing.

	Default: 0.2				Default: 0.4				Default: 0.6			
	Micro		Macro		Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.04	.87	.01	.81	.04	.88	.01	.82	.04	.88	.01	.82
0.10	.08	.88	.03	.86	.09	.89	.03	.86	.09	.89	.04	.86
0.20	.18	.87	.09	.80	.18	.87	.09	.81	.18	.88	.09	.81
0.40	.34	.81	.20	.66	.34	.81	.20	.67	.34	.82	.20	.67
0.80	.59	.69	.34	.51	.59	.69	.34	.50	.59	.69	.33	.50
1.60	.78	.46	.49	.30	.78	.45	.49	.30	.78	.45	.49	.30
3.20	.88	.26	.64	.21	.88	.26	.64	.21	.88	.26	.64	.21
6.40	.94	.15	.75	.10	.94	.15	.75	.10	.94	.15	.75	.10
12.80	.97	.09	.80	.05	.97	.09	.79	.05	.97	.09	.80	.05
25.60	.98	.06	.86	.03	.98	.06	.86	.03	.98	.06	.86	.03
51.20	.99	.03	.89	.02	.99	.03	.89	.02	.99	.03	.89	.02
102.40	.99	.02	.89	.01	.99	.02	.89	.01	.99	.02	.89	.01
B.E.	.64		.41		.63		.41		.63		.41	

Table 9.4 Effectiveness of proportional assignment applied to unscaled category assignment probability estimates. Estimates were generated by Fuhr's classification formula. Results are presented for 10, 30, and 90 features per category.

	10 Features				30 Features				90 Features			
	Micro		Macro		Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.04	.89	.01	.80	.04	.89	.01	.83	.04	.88	.01	.80
0.10	.09	.89	.04	.87	.08	.86	.03	.81	.09	.87	.04	.82
0.20	.18	.86	.10	.81	.17	.84	.09	.79	.17	.83	.09	.74
0.40	.34	.81	.20	.71	.34	.81	.20	.67	.32	.77	.17	.57
0.80	.59	.70	.35	.54	.56	.66	.31	.48	.56	.66	.28	.44
1.60	.79	.46	.52	.33	.78	.46	.47	.30	.77	.45	.43	.27
3.20	.91	.27	.70	.22	.91	.26	.64	.20	.88	.26	.58	.18
6.40	.95	.15	.76	.10	.94	.15	.75	.10	.95	.15	.69	.09
12.80	.97	.09	.81	.06	.98	.09	.83	.05	.98	.09	.79	.05
25.60	.99	.06	.85	.03	.98	.06	.86	.03	.99	.06	.85	.03
51.20	.99	.03	.88	.02	.99	.03	.91	.02	.99	.03	.88	.02
102.40	.99	.02	.91	.01	1.00	.02	.92	.01	1.00	.02	.92	.01
B.E.	.64		.44		.61		.39		.61		.36	

Table 9.5 Effectiveness of proportional assignment applied to unscaled category assignment probability estimates. Estimates were generated by Fuhr's classification formula. Results are presented for 1, 4, and 7 features per category.

	1 Feature				4 Features				7 Features			
	Micro		Macro		Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.03	.67	.01	.57	.04	.91	.01	.85	.04	.88	.01	.81
0.10	.06	.66	.03	.55	.09	.87	.03	.80	.09	.88	.03	.82
0.20	.14	.68	.06	.52	.18	.86	.10	.81	.18	.87	.09	.79
0.30	.21	.67	.10	.53	.26	.82	.15	.75	.26	.83	.15	.75
0.40	.27	.64	.14	.49	.34	.81	.20	.68	.34	.82	.20	.70
0.50	.31	.59	.17	.45	.42	.79	.25	.66	.42	.79	.25	.65
0.60	.35	.56	.19	.42	.49	.77	.29	.63	.49	.77	.28	.61
0.70	.40	.54	.23	.40	.54	.73	.33	.57	.54	.73	.32	.56
0.80	.44	.51	.25	.39	.60	.71	.36	.55	.59	.70	.36	.54
0.90	.48	.50	.27	.34	.62	.65	.38	.48	.62	.65	.38	.47
1.00	.51	.48	.29	.32	.65	.61	.41	.46	.65	.61	.41	.46
1.25	.56	.42	.34	.31	.72	.54	.48	.42	.72	.54	.46	.41
1.50	.59	.37	.40	.26	.75	.47	.55	.35	.77	.48	.51	.34
1.75	.62	.33	.43	.25	.77	.41	.57	.32	.80	.42	.55	.31
2.00	.65	.30	.46	.23	.79	.37	.59	.29	.82	.38	.59	.30
3.00	.72	.22	.55	.18	.85	.27	.68	.23	.89	.28	.67	.23
4.00	.75	.18	.59	.16	.87	.20	.71	.19	.91	.21	.70	.19
16.00	.89	.07	.66	.04	.96	.08	.79	.04	.97	.08	.81	.04
128.00	.97	.02	.74	.01	.99	.02	.86	.01	.99	.02	.89	.10
B.E.	.49		.32		.63		.44		.63		.44	

Table 9.6 Effectiveness of proportional assignment applied to unscaled category assignment probability estimates. Estimates were generated by Fuhr's classification formula. Results are presented for 10 and 15 features per category.

	10 Features				15 Features			
	Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.04	.89	.01	.80	.04	.84	.01	.73
0.10	.09	.89	.04	.87	.08	.86	.03	.83
0.20	.18	.86	.10	.81	.18	.85	.09	.80
0.30	.26	.84	.15	.77	.26	.82	.15	.79
0.40	.34	.81	.20	.71	.34	.80	.20	.71
0.50	.42	.80	.25	.65	.41	.78	.24	.65
0.60	.49	.77	.28	.62	.48	.76	.28	.62
0.70	.55	.74	.32	.58	.54	.73	.33	.58
0.80	.59	.70	.35	.54	.58	.68	.35	.54
0.90	.64	.67	.38	.48	.62	.65	.38	.48
1.00	.67	.63	.40	.46	.65	.61	.40	.46
1.25	.72	.54	.44	.41	.72	.54	.45	.41
1.50	.78	.49	.51	.34	.77	.48	.49	.32
1.75	.81	.43	.53	.31	.80	.43	.51	.28
2.00	.83	.39	.58	.29	.83	.39	.56	.27
3.00	.90	.28	.68	.24	.90	.28	.67	.23
4.00	.93	.22	.72	.19	.93	.22	.74	.19
16.00	.97	.08	.82	.05	.98	.08	.86	.05
128.00	.99	.02	.91	.01	.99	.02	.92	.01
B.E.	.65		.43		.63		.43	

Table 9.7 Breakeven points for five categorization methods. (Microaveraged breakeven point for k-per-doc is extrapolated rather than interpolated.) 10 features are used in all cases.

	Breakeven Points	
	Micro	Macro
Proportional assignment; unscaled estimates	.65	.43
Proportional assignment; scaled estimates	.62	.41
Probability thresholding; unscaled estimates	.59	.46
Probability thresholding; scaled estimates	.61	.44
k-per-doc; unscaled (scaled is same) estimates	.55	.42

five methods, while the complete results can be seen in column 1 of Table 9.6, along with Tables 9.8, 9.9, and 9.10.

Finally, it occurred to us that even though k-per-doc was considerably worse than the other two methods, information about the ranking of categories with respect to documents (k-per-doc) might be combined with information about ranking of documents with respect to categories (proportional assignment). We made a small investigation of this by intersecting the binary categorization produced by proportional assignment at various levels with the binary categorization produced by k-per-doc at various levels. We show the results (microaveraged scores only) in Table 9.11.

The pattern in the data is made more clear in Table 9.12 by presenting only the best recall precision pairs. The values in Table 9.12 are those from Table 9.11 that are not exceeded in both recall and precision by some other value in Table 9.11, or in the corresponding table (not shown) for 6-per-doc through 10-per-doc. This reveals a pattern where the most effective intersections are between categorization strategies with comparable precision and recall. The intersection strategies are always superior to k-per-doc, but are superior to pure proportional assignment only at high precision levels.

9.4 Analysis

The best breakeven point between microaveraged recall and precision for the experiments in this chapter was 0.65 vs. a breakeven point of 0.44 (using the Carnegie Group test set) in the preceding chapter. While still falling well short of the effectiveness levels of the CONSTRUE system on the Carnegie Group test set, this level of effectiveness appears to be as good as or better than that achieved in previous studies on statistical text categorization. As a point of comparison, the AIR/X system, which is the result of a large-scale effort to provide automated indexing as an aid to manual indexing, uses both rule-based and statistical techniques to achieve microaveraged recall and precision of approximately 0.65 for a set of approximately 23,000 categories [FHL+91].

Table 9.8 Probability thresholding on unscaled and scaled scores. 10 features.

	Unscaled Scores				Scaled Scores			
	Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.04	.78	.10	.83	.05	.80	.10	.83
0.10	.08	.77	.14	.75	.09	.83	.12	.81
0.20	.18	.83	.16	.73	.21	.86	.16	.74
0.30	.28	.82	.20	.69	.28	.86	.19	.73
0.40	.36	.80	.26	.66	.36	.83	.23	.71
0.50	.40	.75	.29	.64	.42	.78	.26	.69
0.60	.44	.69	.32	.57	.47	.74	.28	.64
0.70	.50	.66	.35	.52	.54	.68	.30	.62
0.80	.54	.63	.40	.52	.57	.63	.31	.61
0.90	.58	.60	.43	.51	.60	.62	.32	.60
1.00	.62	.58	.45	.47	.62	.58	.32	.59
1.25	.71	.53	.49	.45	.66	.40	.34	.56
1.50	.75	.46	.52	.42	.66	.40	.34	.56
1.75	.79	.42	.54	.39	.71	.38	.38	.53
2.00	.82	.37	.55	.37	.73	.34	.41	.49
3.00	.87	.26	.58	.33	.79	.24	.47	.38
4.00	.92	.21	.67	.27	.81	.18	.49	.34
16.00	.98	.05	.75	.15	.90	.05	.63	.19
128.00	1.00	.01	.87	.03	.99	.01	.84	.02
Breakeven:	.59		.46		.61		.44	

A number of factors may have contributed to the increase in effectiveness over our results in the preceding chapter: the introduction of mutual information for feature selection, use of category specific feature sets, use of smaller feature sets, and a better Bayesian estimation model. A test corpus where the distribution of categories was more similar to the training corpus undoubtedly helped as well. For the purposes of this study we will not attempt to tease out which of these factors was most important, since our main interest is in developing a categorization method effective enough to be a reasonable means of comparing text representations.

The lack of impact of probabilistic indexing was quite surprising, and we checked these results thoroughly to verify they were correct. Several explanations are possible. The mutual information measure used for feature selection was based on binary (nonprobabilistic) indexing, and so may have chosen features that were not only of high quality, but where any occurrence of the word in text was significant. The particular probabilistic indexing formula, while effective in the inference network retrieval model, may not be appropriate for the more traditional Bayesian formula we used. Another factor to consider is that most experiments on probabilistic indexing have assumed stemming of words, which results in higher within document

Table 9.9 Effectiveness of k-per-document using unscaled scores and 10 features. (Microaveraged breakeven point is extrapolated rather than interpolated.)

	Microaveraged		Macroaveraged	
No. Per Doc	Recall	Precision	Recall	Precision
1	.66	.40	.33	.57
2	.78	.24	.44	.37
3	.84	.17	.50	.31
4	.89	.14	.54	.26
5	.92	.11	.58	.25
6	.94	.10	.61	.22
7	.95	.08	.64	.21
8	.96	.07	.66	.20
9	.96	.07	.68	.19
10	.97	.06	.69	.18
11	.97	.05	.71	.17
12	.98	.05	.72	.17
13	.98	.05	.73	.16
14	.98	.04	.73	.16
15	.98	.04	.74	.15
16	.98	.04	.74	.15
17	.98	.04	.75	.15
18	.99	.03	.76	.14
19	.99	.03	.76	.14
20	.99	.03	.77	.14
Breakeven:	.55		.42	

Table 9.10 Proportional assignment using scaled scores and 10 features.

	Microaveraged		Macroaveraged	
No. Per Doc	Recall	Precision	Recall	Precision
0.05	.04	.87	.01	.75
0.10	.09	.89	.03	.85
0.20	.18	.87	.10	.84
0.30	.26	.84	.15	.77
0.40	.34	.81	.20	.69
0.50	.41	.78	.24	.62
0.60	.48	.76	.28	.60
0.70	.53	.72	.31	.55
0.80	.57	.68	.33	.50
0.90	.61	.64	.35	.44
1.00	.63	.60	.38	.42
1.25	.68	.51	.44	.39
1.50	.72	.45	.49	.32
1.75	.75	.40	.51	.29
2.00	.76	.35	.53	.27
3.00	.80	.25	.58	.20
4.00	.83	.19	.60	.16
16.00	.90	.07	.68	.04
128.00	.97	.02	.80	.01
Breakeven:	.62		.41	

frequencies. Finally, the proportional assignment strategy may wash out subtle differences in category/document scores.

The improvement in the effectiveness of proportional assignment relative to the other category assignment strategies was expected given the change in test set. However, it should be noted that Formula 9.15 explicitly includes the prior probability $P(C_j = 1)$, and thus supposedly takes into the relative frequency of the categories. The need to take this frequency into account again in binary categorization points out again that considerable improvement could be made in our probability estimation methods.

9.5 Summary

The experiments reported in this chapter attempted to compensate for a number of deficiencies encountered in the preceding chapter. Foremost among the changes were correcting an unrepresentative training/test set division and introducing a much more powerful feature selection method. Several enhancements were also made to the formula used to estimate the probability of category assignment. The result was significantly improved effectiveness. Our best performing method involves:

Table 9.11 Proportional assignment intersected with k-per-doc, using unscaled scores and 10 features. Only microaveraged scores are presented. Along the left edge and top we show the recall and precision of the individual strategies being intersected.

		1 Per Doc (.66 .40)		2 Per Doc (.78 .24)		3 Per Doc (.84 .17)		4 Per Doc (.89 .14)		5 Per Doc (.92 .11)	
		Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	(.04 .89)	.04	.90	.04	.89	.04	.88	.04	.89	.04	.89
0.10	(.09 .89)	.08	.90	.08	.88	.08	.88	.08	.88	.09	.88
0.20	(.18 .86)	.15	.90	.17	.87	.17	.86	.17	.86	.17	.86
0.30	(.26 .84)	.22	.88	.24	.85	.25	.84	.26	.84	.26	.84
0.40	(.34 .81)	.29	.87	.31	.83	.33	.82	.33	.81	.33	.81
0.50	(.42 .80)	.35	.86	.39	.82	.40	.80	.41	.80	.41	.80
0.60	(.49 .77)	.41	.84	.45	.80	.47	.79	.48	.78	.48	.78
0.70	(.55 .74)	.45	.81	.50	.77	.52	.76	.53	.74	.54	.74
0.80	(.59 .70)	.49	.78	.54	.74	.56	.72	.57	.71	.58	.70
0.90	(.64 .67)	.52	.75	.58	.71	.60	.69	.61	.68	.62	.67
1.00	(.67 .63)	.54	.71	.60	.67	.63	.66	.64	.64	.65	.64
1.25	(.72 .54)	.56	.66	.64	.60	.67	.58	.69	.56	.70	.55
1.50	(.78 .49)	.60	.64	.68	.57	.72	.54	.74	.52	.76	.51
1.75	(.81 .43)	.61	.62	.70	.52	.74	.49	.76	.47	.78	.46
2.00	(.83 .39)	.62	.59	.71	.49	.76	.45	.78	.43	.80	.42
3.00	(.90 .28)	.64	.52	.74	.42	.80	.38	.83	.35	.86	.33
4.00	(.93 .22)	.65	.48	.76	.37	.82	.34	.85	.30	.88	.27
16.00	(.97 .08)	.66	.40	.78	.28	.84	.22	.88	.18	.92	.16
128.00	(.99 .02)	.66	.40	.78	.24	.84	.17	.89	.14	.92	.11
B.E.	.65	.61		.63		.64		.64		.65	

Table 9.12 Proportional assignment intersected with k-per-doc using unscaled scores and 10 features. Only nicroaveraged scores are presented. Scores are omitted if the score of some other intersection of proportional assignment and a run from any of 1-per-doc through 10-per-doc is superior on both recall and precision.

			1 Per Doc ( )		2 Per Doc ( )		3 Per Doc ( )		4 Per Doc ( )		5 Per Doc ( )	
			Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	(	)	.04	.90								
0.10	(	)	.08	.90								
0.20	(	)	.15	.90								
0.30	(	)	.22	.88								
0.40	(	)	.29	.87								
0.50	(	)	.35	.86								
0.60	(	)	.41	.84			.47	.79				
0.70	(.55	.74)	.45	.81	.50	.77	.52	.76				
0.80	(.59	.70)	.49	.78			.56	.72	.57	.71		
0.90	(.64	.67)			.58	.71	.60	.69	.61	.68		
1.00	(.67	.63)					.63	.66			.65	.64
1.25	(.72	.54)					.67	.58	.69	.56	.70	.55
1.50	(.78	.49)			.68	.57	.72	.54	.74	.52	.76	.51
1.75	(.81	.43)										
2.00	(.83	.39)										
3.00	(.90	.28)									.86	.33
4.00	(.93	.22)										
16.00	(.97	.08)										
128.00	(.99	.02)										

- Selection of 10 predictor features per category by mutual information.
- Probability estimates generated by Fuhr's Bayesian formula with binary indexing.
- Proportional assignment of categories with no scaling of probability estimates.

This method appears to be an appropriate testbed for our work on comparing text representations. We therefore return to our hypothesis about syntactic phrase clustering in the next chapter.

CHAPTER 10

TEXT CATEGORIZATION: PHRASE CLUSTERING EXPERIMENTS

As we discussed in Chapter 6, the task of text categorization has many advantages for studying text representations. The past three chapters have described the development and testing of an effective text categorization method and associated software. With these in hand, we returned to our hypotheses that traditional term clustering methods would improve the effectiveness of a text representation based on syntactic indexing phrases.

We first explored this hypothesis, via a text retrieval test collection, in Chapter 5. We found minor improvements in text retrieval effectiveness when combining phrases and phrase clusters with words as a text representation, vs. combining only phrases with words. The large number of confounding factors made it difficult to draw solid conclusions from that experiment, however.

The text categorization task made possible more careful analysis of the effects of text representation choices than did the text retrieval task. We begin this chapter by discussing the kinds of hypotheses that can be posed about text categorization effectiveness using different text representations. We then discuss the changes made in our clustering strategy and in our formation of syntactic indexing phrases, in order to control for confounding factors. Finally we decompose our original hypothesis into a number of finer-grained hypotheses.

We then present our experimental results. Strong support was found for a number of our minor hypotheses. However, our major result was quite unexpected: in this carefully controlled experiment, traditional term clustering resulted in no change in the quality of a syntactic indexing phrase representation. We analyze the reasons for this result and draw some conclusions about approaches to term clustering.

10.1 Comparing Text Representations Via Text Categorization

The form of the hypothesis tested in our experiments with the CACM collection was:

- A text retrieval system will exhibit higher effectiveness when documents are represented using representation X than when they are represented using representation Y.

This hypothesis was tested by running an IR system on a test collection, and varying the text representation while holding all other parameters constant to the extent possible. The text representations investigated were words, and words combined with phrases, phrase clusters, or both. A major difficulty in interpreting the results was the need to choose the set of features for each class (i.e. user information need) on the basis of a small textual user request, which was not written with knowledge of the various text representations it would be used with.

We can pose hypotheses similar to the one above about text representation for text categorization systems:

- A text categorization system will exhibit higher effectiveness when documents are represented using representation X than when they are represented using representation Y.

Hypotheses of this form can be tested by computing categorization effectiveness when documents are represented by features from each of the text representations. Feature selection for each category can be done on the basis of a training sample, so the problem of how to decide what features from a text representation are specified by a textual request does not arise.

Text categorization also allows us to test more detailed hypotheses about the properties of a text representation. The interaction of the text representation with the dimensionality of the classifier is of particular interest. Since an automated feature selection method can be set to choose any desired number of features, it is straightforward to test hypotheses such as these:

- A text categorization system will exhibit higher effectiveness when documents are represented by d features selected from representation X than when they are represented by d features from representation Y.

- A text categorization system will exhibit higher effectiveness when documents are represented by c features selected from representation X than when they are represented by d features from representation X.

Such hypotheses would be difficult to test in a text retrieval context, since a small number of fixed user requests do not give much flexibility in varying feature set size.

The question of statistical significance arises again. For text retrieval we mentioned that making claims about the statistical significance of results was difficult, but that some standard guidelines for significance had a certain degree of acceptance among text retrieval researchers.

Similar problems accompany text categorization. The large number of test documents that can be used would aids claims of significance. On the other hand, it is even less plausible that categories are independent samples than that queries are, since the set of categories was created as a whole. In addition, the presence of near duplicate documents means that documents cannot be considered independent samples either.

For text retrieval experiments, a difference of 10% in precision averaged over fixed recall level has come to be considered significant in the absence of useful statistical significance tests, as mentioned in Chapter 5. Unfortunately, no such ad hoc but accepted test exists for text categorization. We will therefore limit ourselves to saying that results provide strong support for or against a hypothesis, or are inconclusive, without making claims of statistical significance.

10.2 Term Clustering Revisited

The switch from text retrieval to text categorization, and the posing of a more detailed set of experimental hypotheses led us to reconsider the clustering method used in Chapter 5. We discuss the choice of a new method below, along with the choice of metafeatures, the choice of the data set to cluster, the handling of unclustered terms, and the defining of values for cluster features.

10.2.1 Clustering Method

In the CACM experiments we produced star clusters of phrases. One potential disadvantage of star clusters is that they are overlapping, i.e. the same phrase can be incorporated in multiple clusters. This means that a certain degree of dependence is inherent in the clustered representation. Dependence between clusters was probably not a significant problem in our text retrieval experiments, since the small number of phrases specified by a user request were unlikely to result in many overlapping clusters. The problem is potentially more severe in text categorization, given our current automated feature selection. If a cluster is selected by mutual information as being highly correlated with a particular category, then other clusters containing some of the same phrases are also likely to be selected. We chose to deal with this by requiring that the clustering method used for our new experiments form nonoverlapping clusters, i.e. that it be a *partitional* method ([JD88], p. 57).

We also wanted the new clustering method to be conservative. By this we meant that if the method put two items into a cluster, a wide variety of other methods should also put those items together in a cluster. While the effectiveness improvements (if any) produced by a conservative method would presumedly be smaller than those produced by a less conservative, but carefully selected, method, negative results were more likely to be meaningful with a conservative method.

We therefore chose to form reciprocal nearest neighbor (RNN) clusters [Mur83]. An RNN cluster consists of two items, each of which is the nearest neighbor of the other according to the similarity metric in use. This is a very conservative cluster definition, since only clusters of size 2 are formed, and not all items are clustered. Note that if two items appear in the same RNN cluster, each would have appeared in the star cluster of which the other was the center.

As is the case with most clustering algorithms, tied similarity values had to be handled as a special case. For RNN clusters the only important ties arise when an item has more than one nearest neighbor. Even then, the only case when a tie needs to be broken arbitrarily is when two or more of the nearest neighbors of X also have

X as one of their nearest neighbors. In this case one of X's nearest neighbors was chosen arbitrarily (by ID number) to form a RNN cluster with X. The two members of the new cluster were then removed from consideration for clustering, but the other nearest neighbors of X were left to participate in other clusters if they had other nearest neighbors. This tiebreaking strategy means that the clustering could be dependent on the order in which items were considered for clustering. However, in the special case where each item had exactly one nearest neighbor, then clustering was order independent.

10.2.2 Metafeatures and Training Data

Clustering features requires not only choosing a clustering method but also defining a set of metafeatures on which the similarity of the features will be judged. In Chapter 5 we used metafeatures corresponding to manually assigned *Computing Reviews* categories. The value of a phrase on a metafeature was the maximum likelihood estimator of the probability that a randomly selected phrase from documents in the specified *CR* category would be the given phrase.

In the CACM experiments, we rejected clustering by cooccurrence, where each metafeature indicates presence or absence in a training document, due to the low frequency of phrases and the fact that we had a small number of relatively short documents. With more and larger documents available in the Reuters collection we decided that clustering by cooccurrence was worth trying. As well as the usual strategy of having treating each document as a binary metafeature, we also tried using the indexing probability from Turtle's formula (Section 9.2.2.4) as the metafeature value.

We still wanted to investigate the use of category-based metafeatures. It was natural to use the several measures of association between categories C_j and terms W_i that had to be computed anyway as part of the Bayesian classification process, in particular $P(C_j = 1|W_i = 1)$, $P(W_i = 1|C_j = 1)$, and the system mutual information $I(W_i; C_j)$. The use of $P(W_i = 1|C_j = 1)$ is similar though not identical to the metafeature value used in the CACM experiments. All of these were investigated as metafeature values for clustering terms, along with mutual information between presence of a term and presence of a category, $I(W_i = 1, C_i = 1)$. As in the CACM experiments we used the cosine correlation to compute similarity between vectors of metafeatures.

It is important to recognize that the category-based metafeatures, however their value is defined, are minor variations on the binary category features we are trying to predict. This means that using the categorization test data in forming term clusters would be a kind of indirect training on the test data, and had to be avoided. Forming clusters on both the training and test data would be less of a problem with document-based metafeatures, but this too could be questioned on the grounds that knowledge of term dependencies might be captured that would not be available in an operational setting. To avoid these problems, all formation of term clusters was done solely on the training set of 14,704 stories.

We call both the document-based metafeatures and the category-based metafeatures *coarse-grained* metafeatures, since they correspond to relatively large bodies of text. We discuss possible fine-grained metafeatures in Chapter 11.

10.2.3 Unclustered Terms

One effect of using a very restrictive clustering method, such as reciprocal nearest neighbor clustering, is that not all terms will be clustered. The question then arises of whether these isolated terms should be left out, so that the representation consists purely of multi-term clusters, or whether they should be treated as one item clusters and retained.

We chose the latter course, since dropping unclustered terms would clearly penalize clustering methods that form fewer clusters. In an informal sense, retaining unclustered terms preserves the information content of the original representation, and lets us examine how effective different methods of packaging that content are.

10.2.4 Feature Values

Since non-binary indexing probabilities were not found to be effective in Chapter 9, we used binary term values for all representations in this chapter. A cluster feature was given a value of 1.0 if either of its member features had a value of 1.0 for that document.

10.3 Syntactic Phrase Indexing Revisited

We also reevaluated the syntactic indexing phrase formation process from the CACM experiments. Syntactic phrase indexing for the CACM experiments was complex, and had a high error rate in terms of syntactically malformed phrases. Whether these errors had a significant impact on effectiveness was unclear, but they were an undesirable confounding factor. It was also unclear whether the broad range of syntactic constructions we were allowing to form phrases might be degrading effectiveness. In this section we describe the new approach we adopted for producing syntactic indexing phrases.

10.3.1 Choosing A Syntactic Indexing Method

What we desired for our new experiments was an approach to phrase formation analogous to reciprocal nearest neighbor clustering; i.e. a phrase formation procedure that produced only phrases that would be produced by almost any reasonable phrase formation procedure. However, since much more variety is present in syntactic analysis strategies and syntactic phrase indexing procedures than in statistical clustering procedures, it is somewhat more difficult to decide what a conservative phrase formation procedure should produce.

Most previous studies of syntactic phrase indexing have extracted phrases only from noun phrases, or only from sequences of adjacent words. Therefore, a reasonably restrictive definition of syntactic indexing phrases is to require them to meet

both constraints, i.e. to require them to be formed from *simple noun phrases*. Simple noun phrases consist only of a head noun and its immediate premodifiers, and so satisfy both conditions.

This restriction was also motivated by the availability of Church's *parts* program, which can isolate simple noun phrases from unrestricted text with high accuracy. We describe the use of this program in the next section.

10.3.2 Extracting Words and Phrases with a Stochastic Tagger

The *parts* program [Chu88] is a stochastic tagger, a class of programs that has recently attracted interest in the computational linguistics community [DeR88, Kup89, deM90, MSW91, BAD91]. These programs accomplish a task similar to that traditionally performed by the preprocessing and morphological analysis components of a syntactic parser—segmenting of a stream of characters into tokens and labeling of these tokens by syntactic class. They differ in making use of stochastic models of probable sequences of word class labels to disambiguate words that are syntactically ambiguous. These stochastic models are typically induced from a large corpus. Morphological knowledge, if any, tends to be encoded in ad hoc rules that match sequences of characters at the end of a word without explicitly decomposing the word into morphemes. A large lexicon, often derived from resources such as a machine-readable dictionary or tagged corpus, is usually used.

In addition to breaking a character stream into word tokens and assigning a syntactic class to each token, *parts* includes a component that attempts to insert brackets at both ends of simple noun phrases. This is accomplished by a stochastic model of word class transitions at the beginning and end of simple noun phrases. We used *parts* to tag and bracket the 22,173 documents in the Reuters corpus. Before using *parts*, we first ran the Reuters collection through a preprocessor that attempted to do the following to each story:

1. Remove the controlled vocabulary indexing and other extraneous material from the beginning of the story. The ID we assigned the story was retained to be used by later processing. A period was put after the ID so it would be treated as a complete sentence by *parts* and so would not affect the processing of the textual portion of the document.

2. Isolate the title of the story, lower case the title except for its first letter, and put a period after the end of the title.

3. Isolate the dateline and byline, if any, and put periods after them.

4. Recognize tabular data and insert a period at the end of each line.

5. Detect paragraph boundaries and insert the end of paragraph marker used by *parts*.

The first four steps were an attempt to put the text in the format that *parts* expects. The original format would not cause *parts* to crash, but would cause

```
^D PATTERN-ID 503 TRAINING-SET
 1-APR-1987 15:54:40.77
TOPICS: groundnut oilseed    END-TOPICS
PLACES: usa    END-PLACES
PEOPLE:     END-PEOPLE
ORGS:     END-ORGS
EXCHANGES:     END-EXCHANGES
COMPANIES:     END-COMPANIES

^E^E^EG^M
^V^V^Af2344^_reute
r f BC-CCC-SELLS-FARMERS-STO   04-01 0080^M
^B^M
CCC SELLS FARMERS STOCK PEANUTS, OFFERS MORE^M
    WASHINGTON, April 1 - The U.S. Commodity Credit Corporation^M
(CCC) sold 6,034 short tons of 1986-crop farmers stock peanuts^M
for domestic crushing, the U.S. Agriculture Department said.^M
    The peanuts were from the Southwest area and were sold at^M
between 8.05 cts per lb (total kernel content), and 11.7225 cts^M
per lb, the department said.^M
    The CCC will offering additional 1986-peanuts for sale at a^M
later date, the department said.^M
 Reuter^M
^C
```

Figure 10.1 A sample Reuters story in raw form, with insertion of identifying information as described in Chapter 8.

its accuracy to decrease, since the program takes into account capitalization and sentence boundaries in assigning parts of speech.[1] Step 4 was also intended to aid the formation of potentially interesting phrases from numeric tables, and to avoid the interpretation of tables as long and peculiar noun phrases. The most important change was lower casing the title, since *parts* uniformly tags strings of upper case letters as proper nouns.

Step 5, on the other hand, was required to avoid exceeding the sizes of internal data structures in *parts*, since some of the Reuters stories were very long. The output of the preprocessing phase for the sample story in Figure 10.1 is shown in Figure 10.2, while the output of *parts* for this story is shown in Figure 10.3.

The *parts* program segments the character string making up a document into tokens before tagging. Since this segmentation was to be used for the phrasal

[1]Robert Krovetz provided us with considerable advice on an appropriate strategy to use here.

```
^D
.End of Discourse
PATTERN-ID-503.
Ccc sells farmers stock peanuts, offers more.
    WASHINGTON, April 1 .  The U.S. Commodity Credit Corporation
(CCC) sold 6,034 short tons of 1986-crop farmers stock peanuts
for domestic crushing, the U.S. Agriculture Department said.
.PP
    The peanuts were from the Southwest area and were sold at
between 8.05 cts per lb (total kernel content), and 11.7225 cts
per lb, the department said.
.PP
    The CCC will offering additional 1986-peanuts for sale at a
later date, the department said.
.PP
 Reuter
^C
```

Figure 10.2 Sample story after preprocessing to prepare it for *parts*.

indexing we chose to use it for the word level indexing as well. According to this segmentation, there were 106,113 distinct word types. To produce a word-based indexing similar to that used in most IR research, we collapsed together types that varied only on syntactic tags or capitalzation, resulting in 80,936 distinct types. We further pruned this set by removing stopwords from the list in Figures 8.2 and 8.3 and removing types that consisted solely of numerals and puncutation. We only used words with a document frequency of at least 2, i.e. that occurred in at least two training documents. We did not use stemming in these experiments, in order to get a more clear picture of the effect of purely statistical clustering on words.

The resulting set of 22,791 words, which we will refer to as WORDS-DF2, was used in the experiments reported in the previous chapter as well as this one. It is interesting to note that of the 80,936 distinct types left after collapsing on tags and capitalization, fully 26,978 consisted purely of numerals and punctuation. Newswire text contains many more numbers than the technical prose found in most IR test collections.

For experiments on clustering words we used stricter conditions, requiring words to occur in at least 5 training documents, and in at most 1,029 training documents (7% of the training documents). The lower bound on document frequency was an attempt to make sure we had a reasonable amount of data on each word, as well as to reduce the total number of words to be clustered, and thus the computational expense. The upper bound on document frequency was motivated by results in previous experiments on word clustering, as surveyed in Section 4.4.1. We call this representation WORDS-DF5-MAX7PCT.

```
.Start of Sentence/.  .End of Discourse/.  .End of Sentence/.

.Start of Sentence/.  [ PATTERN-ID-503/CD ] ./.  .End of Sentence/.

.Start of Sentence/.  [ Ccc/NN ] sells/VBZ [ farmers/NNS stock/NN
peanuts/NNS ] ,/, offers/VBZ [ more/JJ ] ./.  .End of Sentence/.

.Start of Sentence/.  [ WASHINGTON/NP/NP ] ,/, [ April/NP/NP 1/CD ]
./.  .End of Sentence/.

.Start of Sentence/.  .End of Sentence/.  .End of Sentence/.

.Start of Sentence/.  [ The/AT U.S/NP./NP Commodity/NP/NP Credit/NP/NP
Corporation/NP/NP ] (/( [ CCC/NP/NP ] )/) sold/VBD [ 6,034/CD short/JJ
tons/NNS ] of/IN [ 1986-crop/CD farmers/NNS stock/NN peanuts/NNS ]
for/FORIN [ domestic/JJ ] crushing/VBG ,/, [ the/AT U.S/NP./NP
Agriculture/NP/NP Department/NP/NP ] said/SAIDVBD ./.  .PP/.  .End of
Sentence/.

.Start of Sentence/.  [ The/AT peanuts/NNS ] were/BED from/IN [ the/AT
Southwest/JJ area/NN ] and/CC were/BED sold/VBD at/IN between/IN [
8.05/CD cts/NNS ] per/IN [ lb/NN ] (/( [ total/JJ kernel/NN content/NN
] )/) ,/, and/CC [ 11.7225/CD cts/NNS ] per/IN [ lb/NN ] ,/, [ the/AT
department/NN ] said/SAIDVBD ./.  .PP/.  .End of Sentence/.

.Start of Sentence/.  [ The/AT CCC/NP/NP will/NN ] offering/VBG [
additional/JJ 1986-peanuts/CD ] for/FORIN [ sale/NN ] at/IN [ a/AT
later/JJ date/NN ] ,/, [ the/AT department/NN ] said/SAIDVBD ./.
.PP/.  .End of Sentence/.  .Start of Sentence/.  [ Reuter/NN ] .End of
File 1/.  .End of File 2/.  .End of File 3/.
```

Figure 10.3 Sample story after tagging and bracketing by *parts*.

1. Use *parts* to break character stream into tagged tokens. (Result was 116,113 distinct types on the full set of 22,173 Reuters documents.)
2. Collapse word types that vary only on syntactic tags and case of characters. (80,936 distinct types on 22,173 documents.)
3. Remove word types on the stopword list in Figures 8.2 and 8.3.
4. Remove word types consisting solely of numbers and punctuation.
5. Remove word types according to document frequency:
   (a) WORDS-DF2: Remove word types occurring in fewer than 2 of the 14,704 training documents. Final number of features: 22,791.
   (b) WORDS-DF5-MAX7PCT: Remove word types occurring in fewer than 5 of the 14,704 training documents, or more than 7% (1,029) of the training documents. Final number of features: 11,390.

Figure 10.4 Details of word-based representations used in this chapter.

We summarize both word-based representations in Figure 10.4. It should be noted that words in these word-based representations are implemented as clusters of more primitive word types from our base representation. Our base representation preserves all capitalization and syntactic tag distinctions, and includes all token types produced by *parts*. This strategy retains maximum flexibility for exploring representational variants. For instance, if stemming were desired it could be implemented as a further clustering of the primitive types.

A number of techniques for indexing documents on syntactic indexing phrases are possible. Following our strategy of retaining maximum flexibility for future experimentation we treated each distinct sequence of word types found between a pair of brackets as a phrasal type, and also treated each distinct token found outside of brackets as a type. The result was a total of 306,932 types corresponding to phrases or tokens outside of phrases. This gave us a dictionary and transaction file that allowed a variety of phrase definitions to be specified in terms of this maximally specific definition.

The only previous IR research using *parts* was an experiment by Salton and colleagues on back of the book indexing [SZB90]. In preprocessing the phrases produced by *parts*, Salton deleted all phrase elements tagged as function words. We adopted a similar strategy of using the syntactic tags to define a stoplist, rather than our explicit stoplist. Given a high quality tagging, this approach may be preferable, since collapsing on capitalization can cause parts of proper names, in particular, to appear to be stopwords. Salton also deleted words from an unspecified list of low quality content words, but we did not try to reproduce that strategy.

1. Use *parts* to segment the text into tokens and assigns a syntactic class label to each token. Brackets are inserted on either side of sequences of tokens suspected of being simple noun phrases.

2. Create a term for each distinct token found outside of brackets, and each distinct sequence of tokens found inside of brackets. (306,932 phrasal and nonphrasal types on the full set of 22,173 Reuters documents after this stage.)

3. Eliminate all single-word types.

4. Remove from the bracketed phrase all words that are tagged with one of the following syntactic classes: AT, DT, DTI, DTS, DTX, PPO, PPS, PPLS, PPSS, QL, WDT, WRB, WP$, WPS, WPO.

5. Lower case all characters in the string.

6. Replace all tokens with tag CD (numbers) with *NUMBER*.

7. Remove the syntactic class symbols from the string. (189,962 phrasal types on 22,173 documents.)

8. Eliminate phrases with only one token. These previously had been multitoken phrases but had one or more function words removed. (183,338 phrasal on 22,173 documents.)

9. Filter on document frequency.

   (a) PHRASE-DF2: Phrase must occur in two or more of the 14,704 training documents (32,521 phrasal types).

   (b) PHRASE-DF5: Phrase must occur in five or more of the 14,704 training documents (7,209 phrasal types).

Figure 10.5 Details of phrasal representations.

The full strategy we used for forming phrases is presented in Figure 10.5. We began by removing those classes specified in Salton's method, except for numbers. The special handling of numbers was motivated by the fact that of the 306,932 original phrasal types, 82,538 contained at least one token tagged as a number (CD). Furthermore, it was clear that large numbers of phrases differed only in the particular number present. We did not want to simply remove the numbers, since this would reduce many multitoken phrases to a single token, and collapse other phrases with different patterns of number use.

We took the approach of replacing all tokens tagged as numbers with the new token *NUMBER*. This can be viewed as a highly specialized form of term clustering. In Figure 10.6 we give some examples of phrases containing numbers after collapsing,

NUMBER NUMBER dlrs : 1,243
NUMBER acres : 59
NUMBER agreement : 18
NUMBER aircraft : 26 .
NUMBER barrels : 109
NUMBER contracts : 70
NUMBER cts vs loss NUMBER cts : 125
NUMBER dlrs : 2,674
NUMBER dlr tax credit : 44
NUMBER gmt : 90
NUMBER largest : 7
NUMBER mar8 NUMBER NUMBER unch corn may7 NUMBER NUMBER : 3
NUMBER mln : 1,554
NUMBER mln dlrs : 2,319
NUMBER shares : 656
july NUMBER : 89
stop NUMBER pct : 14

Figure 10.6 Some phrases formed by replacing all numbers (tokens tagged CD) with *NUMBER*. Each phrase is followed by the number of distinct phrasal types (out of 306,932) collapsed to this form.

along with the number of variants that were collapsed to them. Some phrase types containing numbers appeared in hundreds or thousands of variant forms, so that collapsing in this fashion yields some potentially very interesting features.

All of the experiments in this chapter were run either with PHRASE-DF2 (phrase must occur in 2 or more training documents) or PHRASE-DF5 (phrase must occur in 5 or more training documents).

10.4 Experimental Hypotheses

The greater flexibility provided by the text categorization task and by our new software made it worthwhile to reconsider the statement of our central hypothesis. We examined whether it could be reformulated as several hypotheses to be investigated in the text categorization task.

Here is a slight restatement of our central hypothesis as introduced in Chapter 4:

- A text representation consisting of clusters of syntactic phrases will allow more effective text classification than a representation consisting of individual words, clusters of words, or individual syntactic phrases.

We suggested in Chapter 4 that the disappointing results of previous work on clustering of words resulted in part from the ambiguity and breadth of meaning of

words, and that clustering of phrases should be more effective. In retrospect, this was also an implicit claim about the effectiveness of a representation using clustered words, so in rephrasing the hypothesis above we make this claim explicit. We did not test the effectiveness of clustered words in Chapter 5, due to limitations of the software available. Manipulation of representations is considerably easier with the Maxcat software, so we will investigate word clustering as well as phrase clustering in this chapter.

The pairs of representations that are most interesting to compare are those that result from alternative choices about the application of transformations to representations. Word-based indexing and syntactic phrase indexing can be viewed as alternative ways of extracting indexing terms from a character stream. Term clustering is a transformation that can be applied to either a word-based or a phrase-based representation. Considering the possible combinations of these transformations yields the following comparisons:

- A. Words *vs.* syntactic indexing phrases
- B. Words *vs.* clusters of words
- C. Syntactic phrases *vs.* syntactic phrase clusters
- D. Words *vs.* syntactic phrase clusters
- E. Word clusters *vs.* syntactic phrases
- F. Word clusters *vs.* syntactic phrase clusters

We consider each of these comparisons in turn, and present hypotheses of the forms discussed in Section 10.1. We also consider what properties we expect of representations in general.

- Hypothesis A1: The optimal effectiveness of a text representation based on using simple noun phrases as indexing phrases will be less than that of a word-based representation.

Interestingly, there has been little attempt in the past to measure the effectiveness of a purely phrasal representation. The obvious reason for this is that almost all studies of text representation have been done on the text retrieval task, where the combination of short user requests and the low frequency of indexing phrases would guarantee poor effectiveness.

We ran one preliminary experiment of this form early in the research reported in Chapter 5. We compared syntactic indexing phrases with clusters of those phrases, and a combination of phrases and phrase clusters. The clustering method used a complex similarity function that was later abandoned. The similarity function gave 90% weight to metafeatures based on *Computing Reviews* categories, and 10% weight to metafeatures based on morphological heuristics. Star clusters of size 8 were formed. The results are shown in Table 10.1, and compared with effectiveness on the

Table 10.1 Comparison of purely phrasal representations with word-based representations. Mean precision values over 50 queries and 10 recall levels for CACM collection.

Representation	Mean Precision
Phrases	5.0%
Phrase Clusters (special, size 8)	6.4%
Phrases + Phrase Clusters (special, size 8)	7.3%
Words	23.8%
Words + Best Phrase Clusters	24.3%
Words + Phrases	24.7%
Words + Phrases + Best Phrase Clusters	25.3%

same request set and test collection for the best word-based representations explored in Chapter 5. Effectiveness of the purely phrasal representations is decidedly inferior.

There are two additional hypotheses that are motivated by factors other than effectiveness of phrases in text retrieval experiments:

- Hypothesis A2: A text representation based on using simple noun phrases as indexing phrases will have a higher optimal feature set size than a word-based representation.

- Hypothesis A3: The superiority of a word-based representation over a text representation based on using simple noun phrases as indexing phrases will decrease with increasing feature set size.

The reasoning behind both these hypotheses is that a small number of low frequency features, as we would have with a small number of syntactic indexing phrases, cannot distinguish at all among many documents, and therefore cannot distinguish documents belonging to a category from those that do not. Note that this is the behavior predicted for low frequency terms by Salton's term discrimination model.

With respect to clusters of words we hypothesize:

- Hypothesis B1: Optimal effectiveness for a text representation based on RNN clustering of words using coarse-grained metafeatures will be essentially the same as optimal effectiveness for a word-based representation.

This has been the result in numerous text retrieval experiments. Our claim is that this poor effectiveness is due to the poor semantic quality of the resulting clusters, rather than to some factor specific to text retrieval systems. We therefore would expect the same result in text categorization.

We now turn to hypotheses about remedying the statistical problems with syntactic indexing phrases:

- Hypothesis C1: Optimal effectiveness for a text representation based on RNN clustering of simple noun phrases using coarse-grained metafeatures will be higher than optimal effectiveness for a representation based on isolated simple noun phrases.

This is our core hypothesis, that term clustering will to some extent remedy the statistical problems with a syntactic phrase representation. We also make two other hypotheses about the interaction of phrase clustering with dimensionality of the selected feature set.

- Hypothesis C2: Optimal effectiveness for a text representation based on RNN clustering of simple noun phrases using coarse-grained metafeatures will occur at a lower feature set size than for a simple noun phrase representation.

As feature set size increases, the redundancy in the clustered representation will increase faster than redundancy in the unclustered representation.

- Hypothesis C3: The difference in effectiveness between a text representation based on RNN clustering of simple noun phrases using coarse-grained metafeatures and a text representation based on simple noun phrases will be higher at smaller feature set sizes.

The reasoning here is that a given number of syntactic phrase clusters carry more information than the same number of syntactic phrases. However, as the number of features increases, the syntactic phrase representation can make up for this to some extent with more features.

The new version of the hypothesis that was our original motivation for phrase clustering is:

- Hypothesis D1: Optimal effectiveness for a text representation based on RNN clustering of simple noun phrases using coarse-grained metafeatures will be higher than optimal effectiveness for a word-based representation.

From an operational standpoint, this is the most interesting of the hypotheses, since word-based representations are standard for most text classification tasks, and since neither syntactic phrase formation nor term clustering have provided significant improvements over words in the past. This reasoning for this hypothesis is that the high semantic quality produced by going to a phrasal representation is quite important to effectiveness, and that term clustering will be effective in curing the statistical problems that undercut this semantic quality.

Given our hypotheses for A through D, the relationships for E and F follow directly, so we will not discuss them separately. Finally, we make the following additional hypothesis:

- Hypothesis G1: All representations investigated will exhibit a dimensionality peak at a point considerably lower than the full feature set size.

From the standpoint of research on machine learning, such a hypothesis seems almost trivial. The feature sets we will investigate below range in size from 3,735 to 35,521 features. A set of 14,704 training documents would provide very few training instances per parameter, if the full feature sets were used.

Furthermore, the quality of features in these feature sets varies widely, both in an absolute sense (*last* or *two years ago* are likely to be poor features for any category) and relative to categories (*gold* is a much better feature for GOLD than for OIL). Using a feature selection mechanism to rank features with respect to each category means that as the number of features increases, not only is dimensionality increasing but the quality of features is, on the whole, decreasing. Both effects, increasing dimensionality and decreasing quality, suggest that a dimensionality peak will be found.

Finally, our results in Chapter 9 show a dimensionality peak, at a quite small feature set size, for a word-based representation, and we believe that there are no fundamental differences between this representation and ones formed by clustering or natural language processing techniques.

From the standpoint of information retrieval, however, Hypothesis G1 is more controversial. As discussed in Chapter 3, there is debate for the case of query expansion during relevance feedback as to whether all or only selected terms from relevant documents should be added. However, in the text categorization task, the factors of user-supplied terms and a high quality training subset are not present, and we believe that the machine learning results are likely to hold.

10.5 Results

All experiments in this chapter make use of the best categorization strategy found in the previous chapters: Fuhr's formula for estimating categorization probabilities, proportional assignment of categories with no scaling of probabilities, and the use of system mutual information for feature selection. Effectiveness measures were computed for the full set of 135 categories. Word-based indexing was used as a baseline text representation. Effectiveness results on the WORDS-DF2 representation from Table 9.4 are repeated as Table 10.2. Effectiveness results for the WORDS-DF5-MAX7PCT representation, which was used as the base representation for word clustering, are presented in Table 10.3.

For clustering experiments, only the PHRASE-DF5 repersentation was used, to ensure that there was a reasonable amount of evidence about the distribution of the phrase, and to reduce the computational expense of clustering. Tables 10.4 and 10.5 give results on using unclustered PHRASE-DF2 and PHRASE-DF5 phrases as features. PHRASE-DF5 is slightly inferior to PHRASE-DF2, but the difference is not large in most cases, so figures for PHRASE-DF5 will be used in most of our comparisons.

We proposed a number of different metafeature definitions for term clustering. Three were tested for clustering WORDS-DF5-MAX7PCT words: 14,704 binary document occurrence metafeatures, 14,704 Turtle weighted document occurrence

Table 10.2 Categorization effectiveness with feature sets of 10, 30, or 90 words.

	WORDS-DF2 10 Features				WORDS-DF2 30 Features				WORDS-DF2 90 Features			
	Micro		Macro		Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.04	.89	.01	.80	.04	.89	.01	.83	.04	.88	.01	.80
0.10	.09	.89	.04	.87	.08	.86	.03	.81	.09	.87	.04	.82
0.20	.18	.86	.10	.81	.17	.84	.09	.79	.17	.83	.09	.74
0.40	.34	.81	.20	.71	.34	.81	.20	.67	.32	.77	.17	.57
0.80	.59	.70	.35	.54	.56	.66	.31	.48	.56	.66	.28	.44
1.60	.79	.46	.52	.33	.78	.46	.47	.30	.77	.45	.43	.27
3.20	.91	.27	.70	.22	.91	.26	.64	.20	.88	.26	.58	.18
6.40	.95	.15	.76	.10	.94	.15	.75	.10	.95	.15	.69	.09
12.80	.97	.09	.81	.06	.98	.09	.83	.05	.98	.09	.79	.05
25.60	.99	.06	.85	.03	.98	.06	.86	.03	.99	.06	.85	.03
51.20	.99	.03	.88	.02	.99	.03	.91	.02	.99	.03	.88	.02
102.40	.99	.02	.91	.01	1.00	.02	.92	.01	1.00	.02	.92	.01
B.E.	.64		.44		.61		.39		.61		.36	

1. Produce the WORDS-DF5-MAX7PCT representation, as in Figure 10.4 (11,390 terms).

2. Form reciprocal nearest neighbor clusters using cosine correlation.

   (a) WC-BINDOC: Clustering uses 14,704 metafeatures with value 1.0 if word is present in that document, 0.0 otherwise. Result is 1,937 clusters and 7,516 singlets, for a total of 9,453 terms.

   (b) WC-PROBIDX: Clustering uses 14,704 metafeatures with value based on the probabilistic indexing formula 9.21 in Section 9.2.2.4, with default value 0.2. Result is 297 clusters and 10,796 singlets, for a total of 11,093 terms.

   (c) WC-MUTINFO-135: Clustering used 135 features with value equal to mutual information between presence of the word and presence of a manual indexing category. Result is 1,442 clusters and 8,506 singlets, for a total of 9,948 terms.

Figure 10.7 Details of word cluster representations.

Table 10.3 Effectiveness of text representation consisting of words with document frequencies between 5 and 1,029.

	WORDS-DF5 -MAX7PCT 10 Features			
	Micro		Macro	
	Rec	Pre	Rec	Pre
0.05	.04	.82	.01	.80
0.10	.08	.78	.03	.85
0.20	.15	.75	.09	.80
0.30	.23	.73	.14	.74
0.40	.30	.73	.19	.66
0.50	.38	.72	.23	.62
0.60	.44	.71	.27	.59
0.70	.50	.67	.31	.55
0.80	.56	.66	.34	.53
0.90	.60	.63	.37	.48
1.00	.64	.60	.39	.45
1.10	.67	.57	.42	.43
1.20	.70	.55	.44	.42
1.25	.71	.54	.45	.41
1.30	.72	.52	.46	.40
1.40	.74	.50	.48	.39
1.50	.76	.47	.51	.34
1.75	.79	.42	.54	.30
2.00	.82	.38	.59	.29
3.00	.89	.28	.69	.23
4.00	.91	.21	.73	.19
16.00	.97	.08	.82	.05
128.00	.99	.02	.92	.01
B.E.	.62		.43	

Table 10.4 Feature set is all simple noun phrases extracted by *parts* that, after preprocessing, have 2 or more tokens and occur in 2 or more training documents. Feature set sizes of 10, 30, and 90 are used.

	PHRASE-DF2 10 Features				PHRASE-DF2 30 Features				PHRASE-DF2 90 Features			
	Micro		Macro		Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.04	.93	.01	.86	.04	.94	.01	.86	.04	.93	.01	.85
0.10	.08	.86	.03	.78	.09	.90	.03	.80	.09	.89	.04	.79
0.20	.16	.79	.07	.60	.17	.83	.08	.66	.17	.83	.08	.65
0.30	.22	.72	.10	.51	.25	.79	.12	.59	.25	.80	.12	.55
0.40	.29	.70	.14	.43	.31	.75	.16	.51	.32	.76	.16	.51
0.50	.35	.66	.16	.39	.37	.71	.19	.45	.38	.72	.19	.44
0.60	.40	.62	.18	.35	.43	.68	.21	.41	.44	.69	.21	.41
0.70	.41	.56	.19	.32	.47	.64	.22	.37	.48	.65	.23	.37
0.80	.45	.53	.21	.29	.49	.58	.24	.34	.53	.62	.25	.35
0.90	.47	.49	.21	.25	.52	.55	.26	.30	.55	.57	.27	.31
1.00	.48	.45	.22	.23	.54	.51	.27	.28	.57	.54	.28	.30
1.10	.49	.42	.23	.22	.55	.47	.28	.27	.59	.50	.29	.28
1.25	.51	.38	.25	.20	.57	.42	.30	.25	.61	.46	.30	.25
1.50	.54	.33	.27	.16	.59	.37	.32	.19	.64	.40	.33	.20
1.75	.56	.30	.28	.14	.62	.33	.34	.17	.66	.35	.34	.18
2.00	.58	.27	.28	.13	.64	.30	.35	.16	.68	.32	.36	.16
3.00	.63	.20	.31	.09	.69	.21	.38	.11	.73	.23	.40	.12
4.00	.66	.15	.32	.08	.71	.17	.39	.09	.77	.18	.42	.10
8.00	.75	.10	.38	.04	.80	.10	.44	.04	.84	.11	.48	.05
16.00	.79	.06	.44	.02	.83	.07	.50	.03	.87	.07	.54	.03
128.00	.94	.02	.60	.01	.95	.02	.67	.01	.96	.02	.71	.01
B.E.	.47		.23		.53		.28		.56		.29	

Table 10.5 Feature set is all simple noun phrases extracted by *parts* that, after preprocessing, have 2 or more tokens and occur in 5 or more training documents. Feature set sizes of 10, 30, and 90 are used.

	PHRASE-DF5 10 Features				PHRASE-DF5 30 Features				PHRASE-DF5 90 Features			
	Micro		Macro		Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.04	.93	.01	.86	.04	.93	.01	.83	.04	.91	.01	.80
0.10	.08	.85	.03	.72	.09	.89	.03	.75	.09	.87	.03	.68
0.20	.16	.79	.07	.57	.17	.81	.07	.59	.16	.80	.07	.53
0.30	.22	.72	.10	.47	.24	.77	.10	.51	.24	.77	.09	.42
0.40	.29	.69	.12	.40	.31	.73	.13	.43	.30	.73	.12	.37
0.50	.34	.65	.14	.35	.36	.69	.15	.37	.37	.70	.14	.33
0.60	.39	.62	.16	.32	.42	.66	.16	.33	.42	.67	.16	.31
0.70	.41	.55	.17	.29	.46	.63	.18	.30	.47	.63	.18	.29
0.80	.44	.53	.19	.27	.48	.57	.19	.28	.51	.60	.20	.27
0.90	.47	.49	.20	.23	.51	.53	.22	.26	.53	.55	.22	.25
1.00	.48	.45	.21	.22	.53	.50	.23	.24	.55	.52	.23	.24
1.10	.49	.42	.22	.21	.54	.46	.24	.24	.57	.49	.23	.22
1.25	.51	.38	.23	.19	.56	.42	.26	.22	.60	.45	.25	.21
1.50	.53	.33	.25	.15	.59	.37	.28	.18	.62	.39	.27	.16
1.75	.56	.30	.27	.14	.62	.33	.30	.16	.64	.34	.28	.15
2.00	.58	.27	.27	.13	.64	.30	.31	.14	.66	.31	.30	.13
3.00	.63	.20	.30	.09	.69	.21	.35	.11	.72	.22	.35	.11
4.00	.66	.15	.33	.08	.72	.17	.38	.09	.75	.18	.37	.09
8.00	.75	.10	.37	.04	.80	.11	.45	.04	.84	.11	.46	.04
16.00	.80	.06	.45	.02	.84	.07	.51	.03	.87	.07	.52	.03
128.00	.95	.02	.62	.01	.95	.02	.67	.01	.96	.02	.67	.01
B.E.	.47		.22		.52		.24		.54		.23	

Table 10.6 Effectiveness of word clusters formed using 3 kinds of metafeatures.

	WC-BINDOC 10 features				WC-PROBIDX 10 features				WC-MUTINFO-135 10 features			
	Micro		Macro		Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.04	.86	.01	.75	.04	.86	.01	.80	.04	.87	.01	.78
0.10	.08	.86	.03	.83	.09	.87	.03	.86	.08	.87	.03	.80
0.20	.17	.84	.09	.77	.17	.81	.09	.80	.17	.81	.09	.78
0.30	.24	.76	.14	.73	.24	.78	.14	.73	.25	.80	.14	.74
0.40	.31	.75	.20	.68	.30	.72	.18	.65	.31	.74	.19	.66
0.50	.38	.73	.24	.62	.34	.72	.23	.61	.38	.72	.24	.62
0.60	.45	.71	.27	.59	.45	.71	.27	.59	.45	.71	.28	.59
0.70	.50	.68	.31	.55	.51	.68	.31	.55	.51	.69	.32	.56
0.80	.55	.65	.34	.52	.55	.65	.34	.53	.57	.67	.35	.53
0.90	.58	.61	.37	.46	.60	.63	.37	.47	.62	.65	.38	.48
1.00	.63	.59	.40	.45	.65	.61	.39	.45	.66	.62	.41	.46
1.10	.67	.57	.42	.43	.68	.58	.41	.43	.70	.60	.43	.44
1.20	.72	.57	.44	.42	.70	.55	.43	.41	.72	.57	.44	.42
1.25	.74	.55	.45	.40	.71	.54	.44	.41	.74	.55	.46	.42
1.30	.74	.54	.45	.39	.72	.52	.45	.40	.75	.54	.47	.41
1.40	.76	.51	.47	.38	.74	.50	.47	.38	.76	.51	.49	.39
1.50	.78	.48	.50	.32	.76	.47	.50	.33	.78	.48	.50	.33
1.75	.80	.43	.53	.29	.79	.42	.53	.30	.80	.43	.55	.30
2.00	.83	.39	.56	.27	.82	.38	.58	.28	.83	.39	.58	.28
3.00	.90	.28	.66	.22	.89	.28	.68	.22	.89	.28	.66	.22
4.00	.92	.21	.70	.18	.91	.21	.72	.18	.91	.21	.72	.18
16.00	.97	.08	.83	.05	.97	.08	.82	.50	.97	.08	.83	.05
128.00	.99	.02	.93	.01	.99	.02	.92	.01	.99	.02	.93	.01
B.E.	.60		.43		.62		.42		.64		.43	

metafeatures, and 135 category-based metafeatures with values equal to the mutual information between the assignment of a word and the assignment of a category:

$$\frac{P(W_i = 1, C_j = 1)}{P(W_i = 1) \times P(C_j = 1)}$$

Note this is a different measure than the system mutual information used to pick features. The three representations are summarized in Figure 10.7.

As can be seen in Table 10.6, the three strategies yielded similar results at feature set size 10, and all results were slightly worse than those for unclustered WORDS-DF5-MAX7PCT words with feature set size 10 (Table 10.3).

Finally we experimented with clustered syntactic phrases. Figure 10.8 summarizes the phrase cluster representations produced. Table 10.7 shows results at

Table 10.7 Effectiveness of phrase clusters formed using metafeatures based on binary occurrence in training documents.

	PC-BINDOC 10 features				PC-BINDOC 30 features				PC-BINDOC 90 features			
	Micro		Macro		Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.04	.92	.01	.83	.04	.93	.01	.85	.04	.89	.01	.73
0.10	.08	.84	.03	.71	.09	.90	.03	.76	.08	.84	.03	.59
0.20	.16	.80	.07	.59	.17	.82	.07	.61	.16	.79	.06	.48
0.30	.23	.74	.10	.48	.24	.78	.11	.50	.23	.75	.09	.40
0.40	.29	.70	.12	.39	.30	.73	.14	.43	.30	.72	.12	.37
0.50	.34	.65	.14	.33	.36	.69	.15	.37	.36	.68	.14	.33
0.60	.39	.62	.15	.30	.42	.66	.17	.33	.41	.65	.16	.31
0.70	.41	.55	.17	.28	.47	.63	.18	.29	.46	.62	.17	.27
0.80	.45	.53	.18	.26	.48	.57	.20	.27	.50	.59	.19	.25
0.90	.47	.49	.19	.22	.52	.54	.22	.24	.52	.55	.21	.23
1.00	.48	.45	.20	.21	.53	.50	.22	.22	.54	.51	.22	.22
1.10	.49	.42	.20	.20	.54	.46	.23	.21	.57	.49	.23	.21
1.25	.51	.38	.22	.18	.56	.42	.25	.21	.59	.45	.24	.20
1.50	.53	.33	.24	.15	.59	.37	.27	.16	.62	.39	.26	.16
1.75	.57	.30	.26	.14	.62	.33	.28	.14	.65	.34	.28	.14
2.00	.59	.28	.28	.13	.64	.30	.29	.13	.68	.32	.29	.13
3.00	.64	.20	.30	.09	.70	.21	.34	.11	.72	.22	.33	.10
4.00	.67	.16	.33	.08	.73	.17	.37	.09	.76	.18	.35	.08
8.00	.75	.10	.38	.04	.80	.10	.45	.04	.83	.11	.43	.04
16.00	.80	.06	.45	.02	.83	.07	.49	.03	.87	.07	.51	.03
128.00	.95	.02	.63	.01	.95	.02	.67	.01	.96	.02	.67	.01
B.E.	.47		.20		.52		.22		.53		.22	

feature set sizes 10, 30, and 90 for clustering of syntactic phrases using binary document occurrence as a metafeature. Results are disappointingly similar to those for unclustered phrases.

We examined the PC-BINDOC clusters, as well as clusters formed under the four sets of 135 category-based metafeatures. On inspection the clusters formed under category-based methods appeared to be of lower quality than the PC-BINDOC clusters. We also uncovered a particularly bizarre failure: as noted in Figure 10.8 the use of $P(W = 1|C = 1)$ as a metafeature led to exactly one cluster being formed! We traced this difficulty to the fact that the expected likelihood estimates for $P(C = 1)$ and $P(C = 1, W = 1)$ were almost identical when category C was assigned to no training documents. This resulted in an estimate of $P(W = 1|C = 1)$ that was very close to 1.0 for the 23 categories that had no occurrences on the training set. These were completely useless metafeatures, but a weight of 1.0 was very high in

1. Produce the PHRASE-DF5 representation as described in Figure 10.5 (7209 terms).

2. Form reciprocal nearest neighbor clusters using cosine correlation. (* indicates that effectiveness was tested.):

   (a) *PC-BINDOC: Clustering uses 14,704 metafeatures with value 1.0 if word is present in that document, 0.0 otherwise. Result is 1,193 clusters and 3,232 singlets, for a total of 4,425 terms.

   (b) PC-W-GIVEN-C-135: Clustering uses 135 metafeatures with value equal to our estimate of $P(W = 1|C = 1)$ for phrase W and category C. Result is 1 cluster and 5,616 singlets, for a total of 5,617 terms.

   (c) PC-C-GIVEN-W-135: Clustering uses 135 metafeatures with value equal to our estimate of $P(C = 1|W = 1)$ for phrase W and category C. Result is 963 clusters and 3,692 singlets, for a total of 4,655 terms.

   (d) PC-MUTINFO-135: Clustering uses 135 metafeatures with value equal to our estimate of the mutual information between the events $W = 1$ and $C = 1$ for phrase W and category C. Result is 743 clusters and 4,132 singlets, for a total of 4,875 terms.

   (e) *PC-W-GIVEN-C-44: Clustering uses 44 metafeatures with value equal to our estimate of $P(W = 1|C = 1)$ for phrase W and category C. Result is 1,883 clusters and 1,852 singlets, for a total of 3,735 terms.

   (f) PC-C-GIVEN-W-44: Clustering uses 44 metafeatures with value equal to our estimate of $P(C = 1|W = 1)$ for phrase W and category C. Result is 953 clusters and 3,712 singlets, for a total of 4,665 terms.

   (g) PC-MUTINFO-44: Clustering uses 44 metafeatures with value equal to our estimate of the mutual information between the events $W = 1$ and $C = 1$ for phrase W and category C. Result is 1,198 clusters and 3,222 singlets, for a total of 4,420 terms.

   (h) PC-SYSMUTINFO-44: Clustering uses 44 metafeatures with value equal to our estimate of the system mutual information between W and C for phrase W and category C. Result is 1,630 clusters and 2,358 singlets, for a total of 3,988 terms.

Figure 10.8 Details of phrase cluster representations.

comparison to the more sensible estimates for higher frequency categories, and so dominated the clustering process.

This drove home the point that a category was not going to be a good metafeature unless it had enough occurrences for its relationship with terms to be measured. Categories with no occurrences on the training set were an extreme example of this, but other low frequency categories were also likely to be poor metafeatures. We therefore conducted new experiments using only the set of 44 categories with 20 or more occurrences on the training set as metafeatures. On examination it appeared that the clusters formed were not significantly better than the ones formed with the 135 metafeatures. We nevertheless generated recall precision results on the best looking set of clusters, PC-W-GIVEN-C-44. These results are shown in Table 10.8 and also show essentially no improvement over individual phrases (Table 10.5).

10.6 Analysis

In analyzing the results of these experiments, we examine whether the above results provide evidence for or against each of our hypotheses. As discussed earlier, we will not attempt statistical significance tests, given that lack of independence among categories and among test documents. We then consider what factors appear to have had the main impact on the results, particularly with respect to the effectiveness of phrase clustering. Finally we compare the results on categorization in this chapter with those on text retrieval, discussing the differences between them, and the implications of these differences.

For ease of reference, we repeat the breakeven figures for all representations tested in this chapter in Table 10.9.

10.6.1 Results on Experimental Hypotheses

- Hypothesis A1: The optimal effectiveness of a text representation based on using simple noun phrases as indexing phrases will be less than that of a word-based representation.

There was strong support for this hypothesis. The effectiveness of the phrase-based representations are particularly poor at high recall levels, where very large drops in precision occur relative to equivalent recall with a word-based representation. (Compare Table 10.2 with Table 10.4.) This is not surprising, given the low frequency of most phrases.

- Hypothesis A2: A text representation based on using simple noun phrases as indexing phrases will have a higher optimal feature set size than a word-based representation.

- Hypothesis A3: The superiority of a word-based representation over a text representation based on using simple noun phrases as indexing phrases will decrease with increasing feature set size.

Table 10.8 Effectiveness of phrase clusters formed using only high frequency categories as metafeatures.

	PC-W-GIVEN-C-44 10 features				PC-W-GIVEN-C-44 30 features				PC-W-GIVEN-C-44 90 features			
	Micro		Macro		Micro		Macro		Micro		Macro	
	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre
0.05	.04	.90	.01	.85	.04	.92	.01	.87	.04	.86	.01	.68
0.10	.08	.85	.03	.74	.08	.86	.03	.74	.08	.84	.03	.63
0.20	.17	.82	.07	.58	.17	.81	.07	.58	.16	.78	.06	.49
0.30	.23	.74	.10	.46	.24	.77	.10	.49	.23	.75	.09	.42
0.40	.29	.70	.12	.39	.30	.73	.13	.41	.30	.71	.12	.36
0.50	.35	.66	.14	.34	.36	.69	.15	.36	.36	.68	.14	.33
0.60	.40	.63	.16	.32	.42	.67	.16	.32	.42	.66	.16	.30
0.70	.43	.58	.18	.30	.47	.64	.19	.30	.46	.63	.17	.27
0.80	.46	.54	.19	.28	.50	.59	.20	.28	.50	.59	.18	.25
0.90	.48	.50	.21	.25	.52	.55	.23	.26	.54	.56	.20	.23
1.00	.49	.46	.22	.23	.54	.51	.24	.25	.56	.52	.22	.22
1.10	.50	.43	.23	.22	.55	.47	.25	.23	.58	.49	.23	.21
1.25	.52	.39	.24	.20	.57	.43	.26	.22	.60	.45	.24	.20
1.50	.55	.34	.26	.16	.60	.37	.29	.18	.62	.39	.26	.16
1.75	.58	.31	.28	.15	.62	.33	.31	.16	.64	.34	.28	.14
2.00	.60	.28	.29	.13	.65	.30	.33	.15	.67	.31	.29	.13
3.00	.65	.20	.32	.10	.70	.22	.37	.11	.73	.23	.34	.10
4.00	.67	.16	.36	.08	.72	.17	.40	.09	.76	.18	.37	.09
8.00	.77	.10	.41	.04	.81	.11	.47	.05	.84	.11	.46	.04
16.00	.81	.07	.48	.02	.84	.07	.51	.03	.87	.07	.52	.03
128.00	.95	.02	.65	.01	.96	.02	.69	.01	.96	.02	.69	.01
B.E.	.48		.23		.53		.24		.55		.22	

Table 10.9 Summary of breakeven figures for all representations. Clustered representations are indented.

	10 features		30 features		90 features	
	Micro B.E.	Macro B.E.	Micro B.E.	Macro B.E.	Micro B.E.	Macro B.E.
WORDS-DF2	.64	.44	.61	.39	.61	.36
WORDS-DF5-MAX7PCT	.62	.43	–	–	–	–
WC-BINDOC	.60	.43	–	–	–	–
WC-PROBIDX	.62	.42	–	–	–	–
WC-MUTINFO-135	.64	.43	–	–	–	–
PHRASE-DF2	.47	.23	.53	.28	.56	.29
PHRASE-DF5	.47	.22	.52	.24	.54	.23
PC-BINDOC	.47	.20	.52	.22	.53	.22
PC-W-GIVEN-C-44	.48	.23	.53	.24	.55	.22

There was strong support for both hypotheses. The optimal breakeven point for the word-based representation is in the vicinity of 10 features, while breakeven for the phrase-based representation was still increasing at 90 features. By 90 features, however, precision at the low recall end was diminishing substantially, so this increase would probably not continue much farther. Hypothesis A3 was also strongly supported. It would be expected to follow from A1 and A2 except under odd circumstances.

- Hypothesis B1: Optimal effectiveness for a text representation based on RNN clustering of words using coarse-grained metafeatures will be essentially the same as optimal effectiveness for a word-based representation.

There was strong support for this hypothesis. Data on three variations of word-based clustering at feature set size 10 showed all variations were very close in effectiveness to using individual words. Examples of every 100th cluster from the three variations are shown in Figure 10.9.

The clusters vary widely in quality. Many of them combine concepts that would better be combined in a phrasal concept. Particularly noticable are clustering of pieces from names such as *S.G. Warburg and Co Ltd* or *Little Rock, Ark.* Clustering of pieces of company names with the company codes, such as *<ALD>* for *Allied-Signal Inc*, also occurred. Many of the words in other example clusters, such as *trained, rock, singer, joins, link, proof, wage*, and *forces* are so ambiguous so that the clusters containing them are likely to be poor features.

- Hypothesis C1: Optimal effectiveness for a text representation based on RNN clustering of simple noun phrases using coarse-grained metafeatures will be higher than optimal effectiveness for a representation based on isolated simple noun phrases.

WC-BINDOC	WC-PROBIDX	WC-MUTINFO-135
trained, vice-chairman	*irritants, endeavour*	*proof, probe*
banning, derivative	*refugee, belfast*	*link, cities*
adm, refco	*accessible, classification*	*solidarity, goodwill*
ald, allied-signal		*allowable, audited*
unc, nc		*violate, blasts*
ark, rock		*fierce, world-wide*
polyethylene, plastic		*deliverable, midsession*
bce, bell		*tanjug, yugoslavian*
otto, poehl		*checked, imf/world*
butcher, singer		*confidentiality, responds*
expeditious, dismisses		*influencing, toyko*
colleges, schools		*personal, wage*
korean, korea		*forces, importance*
deducted, recapitalize		*tx, texaco*
substitutes, incur		*sporadic, refusing*
exports, imports		
w.r, gra		
joined, joins		
nikkei, simex		
s.g, warburg		

Figure 10.9 Every 100th cluster from three representations consisting of word clusters and singlets.

There was strong evidence against this hypothesis. Effectiveness using phrase clusters varies little from effectiveness using simple noun phrases. Examples of every 100th cluster from the two phrase cluster variations are shown in Figure 10.10 and Figure 10.11. As can be seen, most of the clusters are of questionable quality.

- Hypothesis C2: Optimal effectiveness for a text representation based on RNN clustering of simple noun phrases using coarse-grained metafeatures will occur at a lower feature set size than for a simple noun phrase representation.

The results did not support this hypothesis. Table 10.9 shows that the optimal macroaveraged breakeven point for both representations occurs with a feature set size in the vicinity of 30 features. We did not reach the optimal microaveraged breakeven point in this set of experiments, but no gross differences in microaveraged effectiveness for the two representations appear in the range of feature set sizes from 10 to 90.

- Hypothesis C3: The difference in effectiveness between a text representation based on RNN clustering of simple noun phrases using coarse-grained metafeatures and a text representation based on simple noun phrases will be higher at smaller feature set sizes.

's investors service inc, march quarter NUMBER
stock split, three-for-two stock split
economic performance, major challenge
NUMBER lows, predatory pricing
par pricing, stock price
substantial new lending, political appeal
austerity program, least NUMBER people
space administration, national aeronautics
co < wfc >, wells fargo
his wife, effective july NUMBER
good news, london today
u.s house, new financing packages

Figure 10.10 Every 100th PC-BINDOC cluster.

's investors service inc, < amo >
NUMBER accounts, state regulators
NUMBER elections, NUMBER engines
NUMBER mln dlr contract, NUMBER working rigs NUMBER year
federal reserve chairman paul volcker, private consumption
additional NUMBER dlrs, america >
baker hughes inc, best interest
canadian bonds, cme board
federal spending, country 's economy
denmark NUMBER, equivalent price
external sector, japan 's gross national product
fuji bank ltd, fund government-approved equity investments
inflation rate, paris club
its share price, new venture
major review, management fee
new policy, representative offices
pay deal, port officials
same-store sales, santa rosa
tax reasons, technical assistance

Figure 10.11 Every 100th PC-W-GIVEN-C-44 cluster.

The results did not support this hypothesis. There is very little difference in the effectiveness of the representations within the range of feature set sizes explored.

- Hypothesis D1: Optimal effectiveness for a text representation based on RNN clustering of simple noun phrases using coarse-grained metafeatures will be higher than optimal effectiveness for a word-based representation.

The results did not support this hypothesis. The best breakeven points found for phrase cluster representations (.55 microaveraged and .24 macroaveraged) were decidely inferior to the best for a word-based representation (.64 microaveraged and .44 macroaveraged). In the following section we will discuss the factors influencing this result.

Finally we have our general hypothesis about the representations used:

- Hypothesis G1: All representations investigated will exhibit a dimensionality peak at a point considerably lower than the full feature set size.

The results strongly support this hypothesis. There appear to be some substantial differences in the locations of the peaks for different representations, but we will not make strong claims about the exact locations of these peaks without further experimentation. Larger feature set sizes are clearly preferable for the phrasal representations, however, which is not surprising given the low document frequency of features in these representations.

10.6.2 Why Did Term Clustering Fail to Produce A Better Representation?

In Section 5.4 we examined why term clustering of syntactic indexing phrases failed to produce a significant improvement in text retrieval effectiveness on the CACM collection. Here we conduct a similar analysis of why term clustering of syntactic indexing phrases failed to produce any improvement in text categorization effectiveness on the Reuters collection.

10.6.2.1 Document Scoring Method

As with the text retrieval task, there are two issues: were the representations used handled appropriately by the learning method, and was the way in which term values were computed appropriate? On both scores the situation for categorization is simpler than for text retrieval. In text retrieval the question of which phrases and which phrase clusters should be included in the classifier (query) was complicated, since the feature set had to be inferred from a textual request. In our categorization experiments, on the other hand, a feature set for each category was selected by a mechanism that treated all text representations equally.

The question of what the value of terms should be for each document was also more simple than in our experiments with the CACM corpus. The clusters we formed in the categorization experiments were symmetric. Since there was no cluster seed,

there was less of an issue of whether the cluster feature should be given a higher weight when one of the cluster members was present than when another member was present. Furthermore, our preliminary experiments with word-based features had not found any advantage to using non-binary feature values, so the obvious strategy was to use binary values for all features.

This is not to say that the appropriate weighting of phrase clusters is well understood. In text retreival, weighting of isolated phrases is still a research issue [CD90, CTL91], and weighting of any form of clusters is even more complex. Appropriate term significance weights in text categorization have hardly been explored at all, and our experiments in Chapter 9 suggest that much remains to be learned about them.

It is possible and perhaps even likely that, if the best method of weighting were known for each of the representations, the difference betweeen a phrase cluster representation and a word-based representation would decrease. However, the large difference between effectiveness of these representations when binary values were used suggests that term values are certainly not the only problem.

10.6.2.2 Statistical Problems

We discussed several potential problems under this heading for text retrieval in Section 5.4. The primary one was lack of training data. In moving to the Reuters corpus, we increased the number of training documents for clustering by a factor of 10 (from 1,425 to 14,704), and the number of phrase occurrences by a factor of 25 (from approximately 61,000 to approximately 1.5 million). Still, it remains the case that the amount of data was insufficient to measure the distributional properties of many phrases encountered.

We are less sanguine than were at the time of the Chapter 5 experiments that acquiring larger and larger corpora will eventually suffice to allow traditional statistical term clustering methods to form appropriate phrase clusters. Evidence suggests that the number of word types used in a corpus grows at a rate faster than the square root of the number of word tokens in the corpus [GC90]. This means that as corpus size grows, the number of pairs of words on which there is inadequate training data increases rather than decreases.

The situation is not quite as bad as this result suggests. Not all pairs of words will be combined to form syntactic indexing phrases. The proportion of syntactic indexing phrase tokens corresponding to syntactic indexing phrase types on which there is little training data will decrease as corpus size grows. However, for reasonable corpus sizes it is likely that the proportion of syntactic indexing phrase tokens corresponding to types with low document frequency will remain substantial. Large amounts of data will always be desirable, but we suspect it will not be sufficient itself to allow useful clustering.

The definition of metafeatures is a key issue to reconsider, given the results of the experiments reported here. Our original reasoning was that, since phrases have such low frequency, we should use metafeatures corresponding to bodies of text large enough that we could expect cooccurrences of phrases within them. The poor quality

of the clusters formed in both the CACM and Reuters experiments suggests that this reasoning is simply wrong. The use of large bodies of text (groups of abstracts corresponding to subject areas for CACM, full text documents or subject groups of such documents for Reuters) gives many opportunities for accidental cooccurrences to arise. A small number of such accidental cooccurrences is sufficient to result in a spuriously strong association, given the low frequency of phrases. In our discussion of future work in the next chapter we discuss alternate approaches.

There is still no reason to believe that the similarity function used (cosine correlation) was significantly worse than other functions that might have been used. However, a more radical shift in the kind of clustering done may be necessary. One difficulty with most conventional clustering methods is that they produce *extensional* cluster definitions. In other words, the clusters they form are defined by giving their extension, a list of exactly what entities are members. The problem with extensional clusters is that new entities—in our case new words and phrases found on test documents but not on training documents—can never be members of clusters. This is a significant gap in the coverage of cluster-based features, particularly phrase clusters.

An alternative is to produce *intensional* cluster descriptions. In this approach, a cluster is defined by specifying properties of the items that belong to the cluster, but not by listing those items explicitly. We did a simple form of intensional clustering in preprocessing phrases. By applying a procedure that removed function words and replaced all numbers with a single type, we created canonical phrases that can be viewed as intensionally defined phrase clusters. Such clusters can implicitly include phrases from unseen test documents. A number of techniques for intensional clustering have been investigated, particularly under the heading *conceptual clustering* [FL85].

A new statistical issue in the categorization experiments was whether feature selection might somehow discriminate against clusters, even if they were of high quality. A more serious worry was that words and phrases that were good features might have distributions that were quite different from those of other terms and be less likely to end up in clusters. An inspection of the clusters selected as features for selected categories suggests that neither of these cases arose. A mixture of clusters and singlets were selected as features, and the clusters that were chosen by feature selection usually included phrases that were chosen as top features from the unclustered representation.

We did notice that our relatively simple approach to feature selection—ranking features by system mutual information and selecting the top k features—led to unnecessarily low quality feature sets in some situations. In Figure 10.12 we show the top 10 features chosen from unclustered phrases for the EARN (corporate earnings) category.

These features look rather odd linguistically. Their choice is explained by the large number of stories similar to the one in Figure 10.13 that appear in the EARN category. (We have edited out control characters from these examples.) The phrases in Figure 10.12 originate as rows of tabular data, and are extremely good clues for stories like the one in Figure 10.13, which almost uniformly belong to the EARN category.

NUMBER cts: 276
NUMBER mths: 18
NUMBER vs NUMBER: 684
div NUMBER cts vs NUMBER cts: 223
net NUMBER vs NUMBER: 1,349
record april NUMBER: 77
record march NUMBER: 48
revs NUMBER mln vs NUMBER mln: 1,407
revs NUMBER vs NUMBER: 521
shr NUMBER cts vs NUMBER cts: 1,093

Figure 10.12 Top 10 phrases selected for the EARN category, and number of variant forms.

However, the EARN category also contains many stories like the ones in Figure 10.14. These stories will not contain many phrases such as those in Figure 10.12. Ranking features on their individual merit and choosing the top k features resulted in a set of features that was redundant and that contained good predictors for only a subset of the members of the category.

This example suggests that using a stepwise feature selection mechanism might significantly improve effectiveness. Note that the feature selection mechanism we did use should have favored the clustered over the unclustered representations, since one goal of term clustering is to reduce redundancy among features. The lack of improvement is further evidence that redundancy was not significantly reduced by term clustering.

10.6.2.3 Syntactic Phrase Formation

Simple noun phrase bracketing by the *parts* program proved to be more accurate than the partial syntactic parsing we attempted in our experiments with the CACM corpus. There were some systematic errors, however, mostly resulting from constructions peculiar to the Reuters stories. The presence of tables of data, and of company codes inserted by Reuters after the names of companies, caused a substantial number of mistakes.

One difference from the CACM experiments was our use of phrases with 3 or more words. The effect of this is hard to gauge. On the positive side, it allowed us to avoid the difficult decision of where to segment multiword compound nominals, and further lessened ambiguity. However, it decreased the already low frequency of phrases, which decreased their effectiveness and ability to be clustered accurately. Also decreasing frequency was the fact that stemming was not used.

Another difference was that only simple noun phrases were formed. It was apparent from the Reuters corpus that verbs carry more information in newswire text than they do in technical abstracts. Verb-noun pairs such as *declare dividend*, *report income*, and *raise rates*, would probably have been good indexing terms,

```
PATTERN-ID 522 TRAINING-SET

TOPICS: earn    END-TOPICS
PLACES: usa    END-PLACES
PEOPLE:     END-PEOPLE
ORGS:     END-ORGS
EXCHANGES:     END-EXCHANGES
COMPANIES:     END-COMPANIES

TRANSDUCER SYSTEMS INC YEAR
    KULPSVILLE, PA, April 1 -
    Shr profit 12 cts vs loss 49 cts
    Net profit 117,000 vs loss 506,000
    Revs 1.1 mln vs 1.2 mln
    Year
    Shr profit seven cts vs loss 89 cts
    Net profit 66,000 vs loss 921,000
    Revs 4.4 mln vs 3.9 mln
    NOTE:1986 reflects tax benefit of 24,000. 1985 reflects tax
benefit of 186,000 for quarter and 573,000 for year.
 Reuter
```

Figure 10.13 A corporate earnings story with tabular data.

to the extent they could be extracted accurately. On the whole, simple noun phrases still appear to be the best content indicators among phrases, and it is doubtful that including indexing phrases from other syntactic constructions would have significantly increased effectiveness of phrase-based representations.

10.6.2.4 Semantic Issues

Restricting phrase formation to simple noun phrases eliminated some of the very low content verb-based phrases that we noted in the CACM collection. There were still a substantial number of relatively poor content indicators, however, including some that were selected as features. It may be worth adopting some form of a stop list to screen these out, though how best to do this for phrases, and whether it can be done in a corpus-independent manner, is unclear.

A potentially more important issue is the possibility of recognizing when structural differences in phrases do not signal important meaning differences. Proper nouns are particularly noticeable case of this. Here are the simple noun phrase types produced for variants of one person's name on the Reuters corpus:

chairman paul volcker
chairman volcker

```
PATTERN-ID 511 TRAINING-SET

TOPICS: earn    END-TOPICS
PLACES: brazil   END-PLACES
PEOPLE:     END-PEOPLE
ORGS:     END-ORGS
EXCHANGES:     END-EXCHANGES
COMPANIES:     END-COMPANIES

******BANKAMERICA SAYS 1ST QTR NET TO BE CUT BY 40 MLN DLRS DUE
TO BRAZILIAN LOANS


PATTERN-ID 516 TRAINING-SET

TOPICS: earn    END-TOPICS
PLACES: usa    END-PLACES
PEOPLE:     END-PEOPLE
ORGS:     END-ORGS
EXCHANGES:     END-EXCHANGES
COMPANIES:     END-COMPANIES

STRAWBRIDGE <STRW> DECLARES STOCK DIVIDEND
    PHILADELPHIA, April 1 - Strawbridge and Clothier said its
board declared a seven pct stock dividend, payable May 14 to
holders of record April 14.
    Earlier, the company reported net income of 20.7 mln dlrs.
 Reuter
```

Figure 10.14 Corporate earnings stories without tabular data.

federal chairman paul volcker
federal reserve board chairman paul volcker
federal reserve chairman paul volcker
federal reserve chief paul volcker
fed 's volcker
fed chairman paul vocker
fed chairman paul volcker
fed chairman volcker
fed chief paul volcker
former federal reserve chairman paul volcker
mr volcker
outgoing u.s federal reserve chairman paul volcker
paul volcker
u.s federal reserve board chairman paul volcker
u.s federal reserve chairman paul volcker
volcker

Ideally, a statistical clustering procedure would detect the relationship between these forms and cluster them together. Realistically, the variants have too low a frequency for this to happen. However, knowledge of the structure of names, along with capacities for spelling correction and abbreviation processing, would allow these phrases to be combined into a single cluster or perhaps a small number of clusters.

There are a wide range of modifiers that make phrases appear quite different, but may not result in large differences in meaning. Figure 10.15 shows a number of phrases that might be grouped into the same cluster (or perhaps a small number of clusters) since the differences in their meanings are unlikely to be significant for many text classification situations. Even more examples with the singular *employee* could be listed.

The usual approach to collapsing such variants is to use a stop list, but this will not always be appropriate. For instance, *company* has very little semantic importance as a modifier of *employees*, but it would make a significant difference in meaning to omit it from a phrase such as *company liabilities*. It seems likely that a more knowledge-intensive approach is necessary to handle such cases. Also, for certain category sets, or certain user requests in a text retrieval setting, we might want the ability to distinguish among some of these concepts. So retaining the original forms as well as forming groups seems desirable.

Sometimes the amount of knowledge needed to form better features is not great. Collapsing of phrases on numbers is one example, and it is possible this procedure could be improved by defining several broad classes of numbers (to distinguish large from small amounts) and by building in knowledge of abbreviations such as *mln* for *million*. Looking at Figure 10.12 suggests another example. If months (*april* and *march* in this example) had been replaced by a standard token, a much more powerful feature would have been produced.

NUMBER employees	*firm 's former employees ?*
NUMBER employes	*former employees*
NUMBER full-time employees	*full-time employees*
NUMBER hourly-rated employees	*his employees*
NUMBER hourly employees	*its employees*
NUMBER more employees	*its hourly employees*
NUMBER new employees	*key employees*
NUMBER other former employees	*many employees*
NUMBER part-time employees	*many existing employees*
NUMBER retired employees	*most employees*
NUMBER salaried employees	*new employees*
company 's NUMBER employees	*our employees*
least NUMBER employees	*plant employees*
least NUMBER full-time employees	*regular employees*
plant 's NUMBER employees	*salaried employees*
plant 's NUMBER hourly employees	*selected employees*
active employees	*senior employees*
certain employees	*staffer employees*
company 's employees	*staff employees*
company employees	*temporary employees*
company employess	*their employees*
current employees	*total employees*
eligible employes	*warehouse employees*

Figure 10.15 Phrases that might usefully be collapsed.

10.6.3 Comparison with Text Retrieval

In Section 5.4 we took the approach of examining reasons why clustering of syntactic phrases provided only a small improvement on the text retrieval task. Given the results of these new experiments, we need to examine the converse: why did phrase clustering provide some improvement on the text retrieval task, when it had no impact on the text categorization task?

This question is difficult to answer. The difference between phrase clusters and phrases was below the level that is considered significant or even noticeable in text retrieval experiments, and so may simply be accidental. However, it is possible that some characteristic of the CACM corpus, such as the technical vocabulary or the smaller documents, led to clusters that were actually better than those formed for the Reuters text. Such a difference was not apparent to our eye, however.

Another possibility is some interaction between phrases and request-based feature selection on the CACM corpus. Recall that in the CACM experiments phrasal representations were always combined with a word-based representation. Because of this, a phrasal representation has the indirect effect of increasing the weight given to those individual words that occur in phrases in the query. They are counted twice for some documents: once as matching a query word, and once as matching a phrase

containing the word. If words that occur in phrases in the query are better features than words that don't occur in phrases, then phrases could increase effectiveness even if they are not themselves good features.

This effect may be enhanced by an anomaly in the way simple noun phrases were treated for the CACM collection. When a simple noun phrase with three or more words was encountered, a syntactic indexing phrase was formed from each combination of a prenominal modifier and the head noun. So the simple noun phrase *very high level languages* would result in the syntactic indexing phrases *very languages*, *high languages*, and *level languages*. If the multi-word simple noun phrase occurred several times on the corpus, then the syntactic indexing phrases resulting from it had a tendency to end up in the same clusters. The result was cluster features that gave a lot of weight to words that were the heads of simple noun phrases, such as *languages* in the above. Again, it is not unreasonable to assume these head nouns are better than average content indicators.

10.7 Summary

This chapter reported results on the effectiveness of text representations based on words and on syntactic phrase indexing, with and without the application of term clustering. We controlled for a number of factors that made the results of our experiments in Chapter 5 difficult to interpret. Textual user requests were replaced by automated feature selection, the need to use both words and phrases was eliminated, and both the syntactic analysis and cluster formation processes were made more conservative and accurate. It should be noted that statistical significance tests were not used, due to depedencies among categories and among documents.

The result was that, contrary to our core hypothesis, statistical feature clustering had almost no impact on the effectiveness of a syntactic indexing phrase representation. Given the controls in our experimental design, we believe that this result will hold up for other syntactic phrase formation methods and other knowledge-free extensional clustering methods, though not necessarily for other metafeature definitions.

An important contributor to this result appears to be, as in the CACM experiments, the relatively small amount of training data in comparison to the low frequency of phrases. The related problem of coarse-grained metafeatures also seems to be an important factor. We discussed reasons to be skeptical that these statistical problems will be solved by moving to even larger corpora, and suggested a shift from extensional to intensional clustering, the introduction of simple domain knowledge, and the use of finer-grained metafeatures as possible solutions.

Our other hypotheses, which were not related to clustering of phrases, were supported. A purely phrasal representation was found to produce lower effectiveness than a word-based representation, and as predicted the phrasal representation improved with higher dimensionality, a phenomenon that it has not been possible to investigate for the text retrieval task. A representation based on word clusters was not found to be more effective than a word-based representation, even though words are less affected by the frequency problems afflicting phrases. Also, representations

were found to exhibit a dimensionality peak at far below full feature set size, as predicted.

CHAPTER 11

CONCLUSIONS

This research was motivated by the paradoxical state of affairs with respect to text representations for text retrieval:

- Text retrieval systems are meant to retrieve categories of documents desired by human beings, yet human labelling of documents with meaningful categories (controlled vocabulary indexing) does not produce more effective text retrieval systems than representing documents by words drawn from raw text.

- Most existing text representations and query languages allow arbitrary subsets of a document database to be retrieved, yet actual retrieval effectiveness is far from perfect.

- The meaning of the language used in a document is crucial to whether a document should be retrieved by a text retrieval system, yet attempts to use natural language processing to produce better representations of document content have so far produced no significant effectiveness improvements.

The state of affairs has been even less clear for other, less-researched text classification tasks such as text categorization and text routing. In this chapter we review the contributions we have made toward improving the state of understanding of text representation for IR. We then discuss directions for future research.

11.1 Contributions

The contributions of this dissertation can be broken down into two groups: those bearing directly on a theory and practice of effective text representation for IR, and those additional technical contributions arising from the experiments pursued in our studies of representation.

With respect to text representation, we had two objectives:

1. To outline a theoretical model that enables the behavior of text classification systems to be in part predicted from measurable characteristics of text representations.

2. To test the predictions of this model on a text representation whose behavior has been difficult to understand under existing models of text classification.

The following contributions were made with respect to the first objective:

- A clarification of the role which classifier formation plays in IR systems. In particular, we have shown how a wide range of IR techniques falling under a variety of headings, including indexing, weighting, clustering, phrase formation, user querying, and automated query expansion are manifestations of strategies for feature selection or feature extraction, when viewed from the standpoint of machine learning. Among other things, this makes clear a number of areas where IR has not taken advantage of research in machine learning, and vice versa.

- The beginnings of a new theoretical model of text classification systems, the concept learning model. The core idea of the CL model is that the machine learning software and human user or knowledge engineer in a text classification system be considered to be components of a single learning mechanism. This composite learning mechanism is viewed as having a hypothesis space search bias that results from the search biases of the individual components. The CL model makes it possible to talk sensibly about differences in the quality of two representations, both of which in theory allow perfect text classification.

- An initial list of desirable properties of text representations, drawn from the machine learning and information retrieval literatures, and applicable under the assumption that the search bias of a composite text classification system is similar to that of traditional machine learning algorithms. This assumption is trivially true when classifier formation is done completely by machine learning (as in some text categorization systems), and becomes a more controversial (but testable) assumption for text classification systems where human influence is of more importance.

- An explanation for the low effectiveness of syntactic phrase indexing in terms of its characteristics as a text representation, and a suggested strategy for feature extraction (term clustering) that is appropriate for improving those characteristics.

With respect to the second objective, we made the following contributions:

- Experimental results on the effectiveness of applying statistical term clustering using coarse-grained metafeatures, to syntactic indexing phrases. Small effectiveness improvements were seen in an experiment on a text retrieval test collection, but did not appear in more carefully controlled experiments on a text categorization test collection. The conservative clustering techniques used, the relatively large amount of training data (in comparison with previous studies), and the minimum of confounding factors in the text categorization task, suggests that this negative result (found with words as well as with syntactic indexing phrases) is likely to hold up for a variety of other approaches to statistical term clustering using coarse-grained metafeatures.

- Experimental results showing that representations based on words and on syntactic indexing phrases have properties, such as dimensionality effects, that are predictable by considering their characteristics as feature sets for classifier formation. This a useful counterbalance to claims that text representations produced by NLP are fundamentally different from traditional IR representations.

- An implemented software system, Maxcat, that allows flexible experimentation with text representation strategies for text classification.

In addition this research makes a number of additional contributions, most notably in the area of text categorization:

- Results, using standard evaluation measures, on the effectiveness of a variety of text categorization algorithms and text representations. Most previous studies of text categorization have tested only a single algorithm, and many have used nonstandard evaluation measures.

- A purely machine-learning-based method for text categorization that achieves, with no human intervention, an effectiveness level comparable to that of some operational text categorization systems.

- A clarification of important problems that remain to be addressed in text categorization, particularly those of feature independence and effective assignment of multiple overlapping categories.

- The uncovering of a number of important pitfalls that can arise in text categorization experiments with large corpora of real-world text, such as near-duplicate documents and time-varying distributions of categories.

- Development of an easily usable test collection for text categorization experiments.

11.2 Future Work

There are a variety of directions in which this research can be pursued further. These again break down into two main areas: text representation for text classification, and various issues associated with text categorization.

With respect to text representation, some directions for future work are:

- Further theoretical development of the CL model, in particular making connections with the existing literature on search behavior by users of text retrieval systems, with linguistic theories of meaning, and with inference-based models of text classification [Tur90].

- Empirical studies on the extent to which the relative effectiveness of text representations is consistent across text classification tasks. For the text retrieval task the specification of text representation features in user requests is a big area for research.

- The use of *fine-grained* metafeatures in term clustering. Several studies in the computational linguistics community have looked at clustering of words based on tendency to occur in the same syntactic frames [HGS75, Hin90, Ino91]. This is a much tighter constraint on the meaning of a linguistic unit than cooccurrence in documents, or even cooccurrence in paragraphs or sentences. The challenge here is whether current syntactic parsing technology can detect these syntactic relationships reliably enough for them to be used as metafeatures.

- As an alternative to treating the syntactic context of terms as a metafeature, using structural clustering algorithms [Tho91]. These would eliminate the somewhat artificial distinction between the object being clustered and its metafeatures.

- The use of simple knowledge bases in term clustering. In Chapter 10, we mentioned several cases where relatively small amounts of knowledge about relationships between words or other linguistic structures would enable substantially better features to be formed. A promising area to explore is combining information from knowledge bases with statistical information about term relationships in a large corpus. "Sloppy" knowledge bases such as thesauri, which do not distinguish cleanly between linguistic structures and their meanings, may be effective in this way when they might not be effective on their own.

  There are a variety of machine learning approaches for using domain knowledge in feature extraction that might also be explored. It is important to recognize that the use of knowledge bases in IR does not require the replacement of well-understood IR techniques with, for instance, theorem provers. The most likely role for knowledge bases in near future IR is in selecting and extracting better features for statistical text classification systems.

- The use of intensional rather than extensional cluster formation, to achieve coverage of unseen phrases.

With respect to text categorization, some of the avenues for further research are:

- We believe that the crucial question in statistical text categorization is how to assign multiple overlapping categories so as to produce for each category a desired recall-precision tradeoff. This problem has been dodged in previous work on probabilistic text categorization, though it appears to be partially addressed by methods that rely on regression rather than on explicit probabilistic models [BFL+88]. Our proportional assignment strategy also addresses this problem to some degree, but is far from perfect, and requires documents to be categorized in batch mode. Techniques that explicitly model the distribution of category scores (as in the Swets model for text retrieval systems [Swe69]) may be a good approach.

- Separate from, but closely related to, the problem of achieving effective assignment of multiple overlapping categories is taking advantage of dependencies between these categories in learning. The fact that certain sets of categories tend to apply to the same objects is an important and underexplored opportunity not only for text categorization but for learning in all intelligent systems.

- The lack of effectiveness of probabilistic indexing in our text categorization experiments was surprising, given that this technique has reliably improved effectiveness in text retrieval. Further investigation using several methods of computing indexing probabilities, as well as several text representations, is needed.

- The use of probabilistic models that assume dependence between indexing terms is likely to be more successful in text categorization than in text retrieval, given the large amount of training data available, so this is a promising area to explore. Machine learning algorithms that allow term dependence in concept descriptions, without necessarily using explicit probability models, should also be explored. The use of stepwise feature selection to reduce feature dependence is another promising approach.

- For applications of text categorization, we may be less interested in the ability of the categorizer to replicate human judgments than in the effectiveness of a system using the categorizer output. The most widely proposed application for text categorization is in indexing documents for text retrieval, so studies of text retrieval effectiveness on categorized text are an obvious step. The fact that nonbinary term values are usually found superior with word-based text representations suggests that binary categorization may not actually be the best approach if categorizer output is meant to support text retrieval.

- The behavior of text categorization systems should be studied at the level of individual categories, as well as the level of overall effectiveness. For instance, we noticed in our experiments that dimensionality peaks occurred at different points for different categories. This was to some degree, but by no means completely, correlated with the number of positive training instances for that category. The partial correlation with sample size suggests that using different feature set sizes for different categories would improve effectiveness, if the proper size can be determined from observable properties of the category, or perhaps by cross-validation.

- More research is needed on how to deal with or take advantage of near-duplicate documents, time-varying category distributions, and other anomalies of real text streams.

Finally, relatively little is known about how to combine knowledge engineering and machine learning approaches in building text classification systems. It appears that very high effectiveness can be achieved with a rule-based approach to text categorization, at the cost of significant human effort. The standard of effectiveness

of systems like CONSTRUE provides a motivating force for work on statistical text categorization, with the goal of reaching similar effectiveness levels. From an application standpoint, however, it is the combination of the two approaches, rather than their competition, which is of interest. For instance, experiments with human feature selection for a statistical categorizer would be of considerable theoretical and practical interest. Techniques for making the maximum use of both machine learning and user knowledge are particularly needed for the text retrieval task, which has the constraints of a limited number of training examples and limited human patience.

BIBLIOGRAPHY

[ALBPR88] Antoniadis, Georges, Lallich-Boidin, Geneviève, Polity, Yolla, and Rouault, Jacques. A French text recognition model for information retrieval system. In *Eleventh International Conference on Research & Development in Information Retrieval*, pages 67–84, 1988.

[And73] Anderberg, Michael R. *Cluster Analysis for Applications.* Academic Press, New York, 1973.

[Atk87] Atkins, Beryl T. Semantic ID tags: Corpus evidence for dictionary senses. In *Proceedings of the Third Annual Conference of the UW Centre for the New Oxford English Dictionary*, pages 17–36, 1987.

[BAD91] Boggess, Lois, Agarwal, Rajeev, and Davis, Ron. Disambiguation of prepositional phrases in automatically labelled technical text. In *AAAI-91*, pages 155–159, 1991.

[Ban71] Banerji, Ranan B. Some linguistic and statistical problems in pattern recognition. *Pattern Recognition*, 3:409–419, 1971.

[BB63] Borko, Harold and Bernick, Myrna. Automatic document classification. *Journal of the Association for Computing Machinery*, pages 151–161, 1963.

[BB64] Borko, Harold and Bernick, Myrna. Automatic document classification. Part II. Additional experiments. *Journal of the Association for Computing Machinery*, 11(2):138–151, April 1964.

[BB87] Boguraev, Bran and Briscoe, Ted. Large lexicons for natural language processing: Utilising the grammar coding system of LDOCE. *Computational Linguistics*, 13(3–4):203–218, 1987. Special Issue on the Lexicon.

[BC64] Box, G. E. P. and Cox, D. R. An analysis of transformation. *Journal of the Royal Statistical Society*, 26(2):211–252, 1964.

[BC87] Belkin, Nicholas J. and Croft, W. Bruce. Retrieval techniques. In Williams, Martha E., editor, *Annual Review of Information Science Technology*, volume 22, pages 110–145. Elsevier Science Publishers, 1987.

[BCL+79] Burnett, John E., Cooper, David, Lynch, Michael F., Willett, Peter, and Wycherley, Maureen. Document retrieval experiments using indexing vocabularies of varying size. I. Variety generation symbols assigned to the front of index terms. *Journal of Documentation*, 35(3):197–206, September 1979.

[Ben82] Ben-Bassat, Moshe. Use of distance measures, information measures and error bounds in feature evaluation. In Krishnaiah, P. R. and Kanal, L. N., editors, *Handbook of Statistics*, pages 773–791. North Holland, Amsterdam, 1982.

[BFL+88] Biebricher, Peter, Fuhr, Norbert, Lustig, Gerhard, Schwantner, Michael, and Knorz, Gerhard. The automatic indexing system AIR/PHYS—from research to application. In *Eleventh International Conference on Research & Development in Information Retrieval*, pages 333—342, 1988.

[BFOS84] Breiman, Leo, Friedman, Jerome H., Olshen, Richard A., and Stone, Charles J., editors. *Classification and Regression Trees.* Wadsworth & Brooks, Pacific Grove, CA, 1984.

[Bha74] Bhattacharyya, K. The effectiveness of natural language in science indexing and retrieval. *Journal of Documentation*, 30(3):235–254, 1974.

[BK82] Buell, Duncan A. and Kraft, Donald H. LIARS: a software environment for testing query processing strategies. In *Conference on Research and Development in Information Retrieval*, pages 20–27, 1982.

[BK85] Boyce, Bert R. and Kraft, Donald H. Principles and theories in information science. In Williams, Martha E., editor, *Annual Review of Information Science and Technology, Volume 20*, chapter 6, pages 153–178. Knowledge Industry Publications, 1985.

[BKZ82] Bollmann, P., Konrad, E., and Zuse, H. FAKYR—a method base system for education and research in information retrieval. In *Conference on Research and Development in Information Retrieval*, pages 13–19, 1982.

[Bor64] Borko, Harold. Measuring the reliability of subject classification by men and machines. *American Documentation*, pages 268–273, October 1964.

[Bow84] Bow, Sing-Tze. *Pattern Recognition: Applications to Large Data-Set Problems.* Marcel Dekker, Inc., New York, 1984.

[Bra71] Brauen, T. L. Document vector modification. In Salton, Gerard, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 24, pages 456–484. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[Bro72] Brookes, B. C. The Shannon model of IR systems. *Journal of Documentation*, 28(2):160–162, June 1972.

[BS74] Bookstein, Abraham and Swanson, Don R. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, pages 312–318, September-October 1974.

[BS75] Bookstein, Abraham and Swanson, Don R. A decision theoretic foundation for indexing. *Journal of the American Society for Information Science*, pages 45–50, January-February 1975.

[BT91] Balcom, Laura Blumer and Tong, Richard M. Advanced Decision Systems: Description of the CODEX system as used for MUC-3. In *Proceedings of the Third Message Understanding Evaluation and Conference*, Los Altos, CA: Morgan Kaufmann, May 1991.

[Buc85] Buckley, Chris. Implementation of the SMART information retrieval system. Technical Report 85-686, Department of Computer Science; Cornell University, 1985.

[Bun90] Buntine, Wray. Myths and legends in learning classification rules. In *AAAI-90*, pages 736–742, 1990.

[CD90] Croft, W. Bruce and Das, Raj. Experiments with query acquisition and use in document retrieval sytems. In *Thirteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 349–365, 1990.

[CG89] Church, Kenneth W. and Gale, William A. Enhanced Good-Turing and Cat-Cal: Two new methods for estimating probabilities of English bigrams. In *Speech and Natural Language Workshop*, pages 82–91, San Mateo, CA: Morgan Kaufmann, October 1989.

[CG91] Church, Kenneth W. and Gale, William A. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5:19–54, 1991.

[CGHH89] Church, Kenneth, Gale, William, Hanks, Patrick, and Hindle, Donald. Parsing, word associations, and typical predicate-argument relations. In *Speech and Natural Language Workshop*, pages 75–81, San Mateo, CA: Morgan Kaufmann, October 1989.

[CH79] Croft, W. B. and Harper, D. J. Using probabilistic models of document retrieval without relevance feedback. *Journal of Documentation*, 35(4):285–295, 1979.

[Cha88] Chatfield, Christopher. *Problem Solving: A Statistician's Guide.* Chapman and Hall, London, 1988.

[Chu88] Church, Kenneth Ward. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing*, pages 136–143, February 1988.

[Cla85] Clancey, William J. Heuristic classification. *Artificial Intelligence*, 27:289–350, 1985.

[Cle91] Cleverdon, Cyril W. The significance of the Cranfield tests of index languages. In *Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1991.

[CLC88] Croft, W. B., Lucia, T. J., and Cohen, P. R. Retrieving documents by plausible inference: A preliminary study. In *Eleventh International Conference on Research & Development in Information Retrieval*, pages 481–494, 1988.

[CM78] Cooper, W. S. and Maron, M. E. Foundations of probabilistic and utility-theoretic indexing. *Journal of the Association for Computing Machinery*, 25(1):67–80, January 1978.

[Col81] Cole, Elliot. Examining Design Assumptions for an Information Retrieval Service: SDI Use for Scientific and Technical Databases. *Journal of the American Society for Information Science*, pages 444–450, November 1981.

[COL83] Cerny, Barbara A., Okseniuk, Anna, and Lawrence, J. Dennis. A fuzzy measure of agreement between machine and manual assignment of documents to subject categories. In *Proceedings of the 46th ASIS Annual Meeting*, page 265, 1983.

[Coo78] Cooper, William S. Indexing documents by gedanken experimentation. *Journal of the American Society for Information Science*, pages 107–119, May 1978.

[Coo91] Cooper, William S. Some inconsistencies and misnomers in probabilistic information retrieval. In *Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 57–61, 1991.

[CR82] Croft, W. Bruce and Ruggles, Lynn. The implementation of a document retrieval system. In *Conference on Research and Development in Information Retrieval*, pages 28–37, 1982.

[Cro81] Croft, W. Bruce. Document representation in probabilistic models of information retrieval. *Journal of the American Society for Information Science*, pages 451–457, November 1981.

[Cro83] Croft, W. B. Experiments with representation in a document retrieval system. *Information Technology: Research and Development*, 2:1–21, 1983.

[Cro87] Croft, W. Bruce. Approaches to intelligent information retrieval. *Information Processing and Management*, 23(4):249–254, 1987.

[Cro88] Crouch, Carolyn J. A cluster-based approach to thesaurus construction. In *Eleventh International Conference on Research & Development in Information Retrieval*, pages 309–320, 1988.

[CTL91] Croft, W. Bruce, Turtle, Howard R., and Lewis, David D. The use of phrases and structured queries in information retrieval. In *Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 32–45, 1991.

[CU91] Callan, James P. and Utgoff, Paul E. A transformational approach to constructive induction. In *Eighth International Workshop on Machine Learning*, pages 122–126, 1991.

[DCR89] Drastal, George, Czako, Gabor, and Raatz, Stan. Induction in an abstraction space: A from of constructive induction. In *IJCAI-89*, pages 708–712, 1989.

[DDF+90] Deerwester, Scott, Dumais, Susan T., Furnas, George W., Landauer, Thomas K., and Harshman, Richard. Indexing by latent semantic indexing. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.

[deM90] de Marcken, Carl G. Parsing the LOB corpus. In *28th Annual Meeting of the Association for Computational Linguistics*, pages 243–251, 1990.

[DeJ82] DeJong, Gerald. An overview of the FRUMP system. In Lehnert, Wendy G. and Ringle, Martin H., editors, *Strategies for Natural Language Processing*, pages 149–176. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1982.

[DeJ88] DeJong, Kenneth. Learning with genetic algorithms: An overview. *Machine Learning*, 3:121–138, 1988.

[DeR88] DeRose, Steven J. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39, 1988.

[DG83] Dillon, Martin and Gray, Ann S. FASIT: A fully automatic syntactically based indexing system. *Journal of the American Society for Information Science*, 34(2):99–108, March 1983.

[DGCN91] Dolan, Charles P., Goldman, Seth R., Cuda, Thomas V., and Nakamura, Alan M. Hughes Trainable Text Skimmer: Description of the TTS system as used for MUC-3. In *Proceedings of the Third Message Understanding Evaluation and Conference*, Los Altos, CA: Morgan Kaufmann, May 1991.

[DH73] Duda, Richard O. and Hart, Peter E. *Pattern Classification and Scene Analysis.* Wiley-Interscience, New York, 1973.

[DHB90] Dietterich, Thomas G., Hild, Hermann, and Bakiri, Ghulum. A comparative study of ID3 and backpropagation for English text-to-speech mapping. In *Seventh International Conference on Machine Learning*, pages 24–31, 1990.

[DLW+91] Dahlgren, Kathleen, Lord, Carol, Wada, Hajime, McDowell, Joyce, and Stabler, Jr., Edward P. ITP Interpretext system: MUC-3 test results and analysis. In *Proceedings of the Third Message Understanding Evaluation and Conference*, Los Altos, CA: Morgan Kaufmann, May 1991.

[DR89] Drastal, G. and Raatz, S. Empirical results on learning in an abstraction space. Technical Report DCS-TR-248, Department of Computer Science; Rutgers University, January 1989.

[DR91] Deogun, Jitender S. and Raghavan, Vijay V. Description of the UNL/USL system used for MUC-3. In *Proceedings of the Third Message Understanding Evaluation and Conference*, Los Altos, CA: Morgan Kaufmann, May 1991.

[Dra91] Drastal, George. Informed pruning in constructive induction. In *Eighth International Workshop on Machine Learning*, pages 132–136, 1991.

[Ear73] Earl, Lois L. Use of word government in resolving syntactic and semantic ambiguities. *Information Storage and Retrieval*, 9:639–664, 1973.

[Fag87] Fagan, Joel L. *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods.* PhD thesis, Department of Computer Science, Cornell University, September 1987.

[Fag89] Fagan, Joel L. The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. *Journal of the American Society for Information Science*, 40(2):115–132, 1989.

[FB90] Fuhr, Norbert and Buckley, Chris. Probabilistic document indexing from relevance feedback data. In *Thirteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 45–61, 1990.

[FH89] Fuhr, Norbert and Huther, Hubert. Optimum probability estimation from empirical distributions. *Information Processing and Management*, pages 493–507, 1989.

[FHL+91] Fuhr, Norbert, Hartmann, Stephan, Lustig, Gerhard, Schwantner, Michael, Tzeras, Konstadinos, and Knorz, Gerhard. AIR/X—a rule-based multistage indexing system for large subject fields. In *RIAO 91 Conference Proceedings: Intelligent Text and Image Handling*, pages 606–623, 1991.

[Fid91] Fidel, Raya. Searcher's selection of search keys. *Journal of the American Society for Information Science*, 42(7):490–527, 1991.

[Fie75] Field, B. J. Towards automatic indexing: Automatic assignment of controlled-language indexing and classification from free indexing. *Journal of Documentation*, 31(4):246–265, December 1975.

[Fis36] Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[FK84] Fuhr, N. and Knorz, G. E. Retrieval test evaluation of a rule based automatic indexing (AIR/PHYS). In van Rijsbergen, C. J., editor, *Research and Development in Information Retrieval: Proceedings of the Third Joint BCS and ACM Symposium*, pages 391–408, Cambridge, July 2-6 1984. Cambridge University Press.

[FK88] Fox, Edward A. and Koll, Matthew B. Practical enhanced boolean retrieval: Experiences with the SMART and SIRE systems. *Information Processing and Management*, 24:257–267, 1988.

[FL85] Fisher, Douglas and Langley, Pat. Approaches to conceptual clustering. In *IJCAI-85*, pages 691–697, 1985.

[FMW71] Friedman, S. R., Maceyak, J. A., and Weiss, S. F. A relevance feedback system based on document transformations. In Salton, Gerard, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 23, pages 447–455. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[FNL88] Fox, Edward A., Nunn, Gary L., and Lee, Whay C. Coefficients for combining concept classes in a collection. In *Eleventh International Conference on Research & Development in Information Retrieval*, pages 291–307, 1988.

[Fox80] Fox, Edward A. Lexical relations: Enhancing effectiveness of information retrieval systems. *SIGIR Forum*, XV(3):5–36, 1980.

[Fox83] Fox, Edward A. Some considerations for implementing the SMART information retrieval system under UNIX. Technical Report 83-560, Department of Computer Science; Cornell University, 1983.

[FU91] Fawcett, Tom E. and Utgoff, Paul E. A hybrid method for feature generation. In *Eighth International Workshop on Machine Learning*, pages 137–141, 1991.

[Fuh89] Fuhr, Norbert. Models for retrieval with probabilistic indexing. *Information Processing and Management*, 25(1):55–72, 1989.

[GB69] Gifford, Carolyn and Baumanis, George J. On understanding user choices: Textual correlates of relevance judgments. *American Documentation*, pages 21–26, January 1969.

[GC90] Gale, William A. and Church, Kenneth W. Poor estimates of context are worse than none. In *Speech and Natural Language Workshop*, pages 283–287, San Mateo, CA: Morgan Kaufmann, June 1990.

[GD78] Goldstein, Matthew and Dillon, William R. *Discrete Discriminant Analysis.* John Wiley & Sons, New York, 1978.

[Gor81] Gordon, A. D. *Classification: Methods for the Exploratory Analysis of Multivariate Data.* Chapman and Hall, London, 1981.

[Gor88a] Gordon, Michael. Probabilistic and genetic algorithms for document retrieval. *Communications of the ACM*, 31(10):1208–1218, October 1988.

[Gor88b] Gordon, Michael D. The necessity for adaptation in modified boolean document retrieval systems. *Information Processing and Management*, 24(3):339–347, 1988.

[GSM91] Grishman, Ralph, Sterling, John, and Macleod, Catherine. New York University description of the PROTEUS system as used for MUC-3. In *Proceedings of the Third Message Understanding Evaluation and Conference*, Los Altos, CA: Morgan Kaufmann, May 1991.

[Ham80] Hamming, Richard W. *Coding and Information Theory.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[Han81] Hand, D. J. *Discrimination and Classification.* John Wiley & Sons, Chichester, 1981.

[HANS90] Hayes, Philip J., Anderson, Peggy M., Nirenburg, Irene B., and Schmandt, Linda M. TCS: A shell for content-based text categorization. In *IEEE Conference on Artificial Intelligence Applications*, 1990.

[Har75a] Harter, Stephen P. A probabilistic approach to automatic keyword indexing. Part II. An algorithm for probabilistic indexing. *Journal of the American Society for Information Science*, pages 280–289, September-October 1975.

[Har75b] Harter, Stephen P. A probabilistic approach to automatic keyword indexing.Part I. On the distribution of specialty words in a technical literature. *Journal of the American Society for Information Science*, pages 197–206, July-August 1975.

[Har82] Harding, P. Automatic indexing and classification for mechanised information retrieval. BLRDD Report No. 5723, British Library R & D Department, London, February 1982.

[Har87] Harman, Donna. A failure analysis on the limitations of suffixing in an online environment. In *Tenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 102–107, 1987.

[Har88a] Hardt, S. L. On recognizing planned deception. In *AAAI-88 Workshop on Plan Recognition*, 1988.

[Har88b] Harman, Donna. Towards interactive query expansion. In *Eleventh International Conference on Research & Development in Information Retrieval*, pages 321–331, 1988.

[Hau88] Haussler, David. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36(2):177–221, September 1988.

[Hay90] Hayes, Philip J. Intelligent high-volume text processing using shallow, domain-specific techniques. In Jacobs, P. S., editor, *Text-Based Intelligent Systems: Current Research in Text Analysis, Information Extraction, and Retrieval*, pages 70–74, Schenectady, NY, 1990. American Association for Artificial Intelligence, General Electric Research and Development Center. Report Number 90CRD198.

[HGS75] Hirschman, Lynette, Grishman, Ralph, and Sager, Naomi. Grammatically-based automatic word class formation. *Information Processing and Management*, 11:39–57, 1975.

[Hin90] Hindle, Donald. Noun classification from predicate-argument structures. In *28th Annual Meeting of the Association for Computational Linguistics*, pages 268–275, 1990.

[Hir87] Hirst, Graeme. *Semantic Interpretation and the Resolution of Ambiguity*. Cambridge University Press, Cambridge, England, 1987.

[HK69] Hillman, Donald J. and Kasarda, Andrew J. The LEADER retrieval system. In *AFIPS Proceedings 34*, pages 447–455, 1969.

[HKC88] Hayes, Philip J., Knecht, Laura E., and Cellio, Monica J. A news story categorization system. In *Second Conference on Applied Natural Language Processing*, pages 9–17, 1988.

[Hob91] Hobbs, Jerry R. SRI International: Description of the TACITUS system as used for MUC-3. In *Proceedings of the Third Message Understanding Evaluation and Conference*, Los Altos, CA: Morgan Kaufmann, May 1991.

[Hor86] Horn, Berthold Klaus Paul. *Robot Vision.* The MIT Press, Cambridge, MA, 1986.

[Hoy73] Hoyle, W. G. Automatic indexing and generation of classification systems by algorithm. *Information Storage and Retrieval*, 9:233–242, 1973.

[Hv78] Harper, D. J. and van Rijsbergen, C. J. An evaluation of feedback in document retrieval using co-occurrence data. *Journal of Documentation*, 34:189–216, 1978.

[HW90] Hayes, Philip J. and Weinstein, Steven P. CONSTRUE/TIS: a system for content-based indexing of a database of news stories. In *Second Annual Conference on Innovative Applications of Artificial Intelligence*, 1990.

[HZ80] Hamill, Karen A. and Zamora, Antonio. The use of titles for automatic document classification. *Journal of the American Society for Information Science*, pages 396–402, 1980.

[Ino91] Inoue, Naomi. Automatic noun classification by using Japanese-English word pairs. In *29th Annual Meeting of the Association for Computational Linguistics*, pages 201–208, 1991.

[Jam85] James, Mike. *Classification Algorithms.* John Wiley & Sons, New York, 1985.

[JC82] Jain, A. K. and Chandrasekaran, B. Dimensionality and sample size considerations in pattern recognition practice. In Krishnaiah, P. R. and Kanal, L. N., editors, *Handbook of Statistics, Vol. 2*, pages 835–855. North-Holland, Amsterdam, 1982.

[JD78] Jain, Anil K. and Dubes, Richard. Feature definition in pattern recognition with small sample size. *Pattern Recognition*, 10:85–97, 1978.

[JD88] Jain, Anil K. and Dubes, Richard C. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[JR90] Jacobs, Paul S. and Rau, Lisa F. SCISOR: Extracting information from on-line news. *Communications of the ACM*, 33(11):88–97, November 1990.

[Jv71] Jardine, N. and van Rijsbergen, C. J. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 7:217–240, 1971.

[KA88] Kibler, Dennis and Aha, David W. Comparing instance-averaging with instance-saving learning algorithms. In *Proceedings of the First International Workshop on Change of Representation and Inductive Bias*, pages 199–211, 1988.

[Kad91] Kadie, Carl Myers. Quantifying the value of constructive induction, knowledge, and noise filtering on inductive learning. In *Eighth International Workshop on Machine Learning*, pages 153–157, 1991.

[KC89] Krovetz, Robert and Croft, W. Bruce. Word sense disambiguation using machine-readable dictionaries. In *Twelfth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 127–136, 1989.

[Kee81] Keen, E. Michael. Laboratory tests of manual systems. In Sparck Jones, Karen, editor, *Information Retrieval Experiment*, chapter 8. Butterworths, London, 1981.

[Kit86] Kittler, J. Feature selection and extraction. In Young, Tzay Y. and Fu, King-Sun, editors, *Handbook of Pattern Recognition and Image Processing*, pages 59–83. Academic Press, Orlando, 1986.

[Kli73] Klingbiel, Paul H. Machine-aided indexing of technical literature. *Information Storage and Retrieval*, 9:79–84, 1973.

[KMT+82] Katzer, J., McGill, M. J., Tessier, J. A., Frakes, W., and DasGupta, P. A study of the overlap among document representations. *Information Technology: Research and Development*, 1:261–274, 1982.

[Kup89] Kupiec, Julian. Augmenting a hidden Markov model for phrase-dependent word tagging. In *Speech and Natural Language Workshop*, pages 92–98, San Mateo, CA, October 1989. DARPA, Morgan Kaufmann.

[KW75] Kar, B. Gautam and White, L. J. A distance measure for automatic sequential document classification. Technical Report OSU-CISRC-TR-75-7, Computer and Information Science Research Center; Ohio State Univ., Columbus, Ohio, August 1975.

[Lac75] Lachenbruch, Peter A. *Discriminant Analysis.* Hafner Press, New York, 1975.

[LC90] Lewis, David D. and Croft, W. Bruce. Term clustering of syntactic phrases. In *Thirteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 385–404, 1990.

[LCB89] Lewis, David D., Croft, W. Bruce, and Bhandaru, Nehru. Language-oriented information retrieval. *International Journal of Intelligent Systems*, 4(3):285–318, 1989.

[Les69] Lesk, M. E. Word-word associations in document retrieval systems. *American Documentation*, pages 27–38, January 1969.

[Lew90] Lewis, David D. A description of CACM-3204-ML1, a test collection for information retrieval and machine learning. Information Retrieval Laboratory Memo 90-1, Computer and Information Science Department, University of Massachusetts at Amherst, 1990.

[Lew91a] Lewis, David D. Data extraction as text categorization: An experiment with the MUC-3 corpus. In *Proceedings of the Third Message Understanding Evaluation and Conference*, Los Altos, CA: Morgan Kaufmann, May 1991.

[Lew91b] Lewis, David D. Learning in intelligent information retrieval. In *Eighth International Workshop on Machine Learning*, pages 235–239, 1991.

[Los88] Losee, Robert M. Parameter estimation for probabilistic document-retrieval models. *Journal of the American Society for Information Science*, 39(1):8–16, 1988.

[LS89] Lochbaum, Karen E. and Streeter, Lynn A. Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval. *Information Processing and Management*, 25(6):665–676, 1989.

[Mar61] Maron, M. E. Automatic indexing: An experimental inquiry. *Journal of the Association for Computing Machinery*, 8:404–417, 1961.

[Mar79] Maron, M. E. Depth of indexing. *Journal of the American Society for Information Science*, pages 224–228, July 1979.

[Mat90] Matheus, Christopher John. Adding domain knowledge to SBL through feature construction. In *AAAI-90*, pages 803–808, 1990.

[Mat91] Matheus, Christopher J. The need for constructive induction. In *Eighth International Workshop on Machine Learning*, pages 173–177, 1991.

[MCM86] Michalski, Ryszard S., Carbonell, Jaime G., and Mitchell, Tom M., editors. *Machine Learning. An Artificial Intelligence Approach. Volume II.* Morgan Kaufmann, Los Altos, CA, 1986.

[Min89] Mingers, John. An empirical comparison of selection methods for decision-tree induction. *Machine Learning*, 3:319–342, 1989.

[Mit82] Mitchell, Tom M. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[MK60] Maron, M. E. and Kuhns, J. L. On relevance, probabilistic indexing, and information retrieval. *Journal of the Association for Computing Machinery*, 7(3):216–244, July 1960.

[MR89] Matheus, Christopher J. and Rendell, Larry A. Constructive induction on decision trees. In *IJCAI-89*, pages 645–650, 1989.

[MSW91] Meteer, Marie, Schwartz, Richard, and Weischedel, Ralph. Studies in part of speech labelling. In *Speech and Natural Language Workshop*, pages 331–336, San Mateo, CA: Morgan Kaufmann, February 1991.

[Mur83] Murtagh, F. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, 1983.

[MWRR86] Mott, Paul M., Waltz, David L., Resnikoff, Howard L., and Robertson, George G. Automatic indexing of text. Technical Report 86-1, Thinking Machines Corporation, January 1986.

[MWZ72] Minker, Jack, Wilson, Gerald A., and Zimmerman, Barbara H. An evaluation of query expansion by the addition of clustered terms for a document retrieval system. *Information Storage and Retrieval*, 8:329–348, 1972.

[NKM77] Noreault, Terry, Koll, Matthew, and McGill, Michael J. Automatic ranked output from boolean searches in SIRE. *Journal of the American Society for Information Science*, pages 333–339, November 1977.

[Pag89] Pagallo, Giulia. Learning DNF by decision trees. In *IJCAI-89*, pages 639–644, 1989.

[Paz90] Pazzani, Michael J. Average case analysis of conjunctive learning algorithms. In *Seventh International Conference on Machine Learning*, pages 339–347, 1990.

[Pis84] Pissanetzky, Sergio. *Sparse Matrix Technology*. Academic Press, Orlando, FL, 1984.

[Por80] Porter, M. F. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[Qui83] Quinlan, J. Ross. Learning efficient classification procedures and their application to chess end games. In Michalski, Carbonell, and Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 463–482. Morgan-Kaufman, Los Altos, California, 1983.

[Qui86a] Quinlan, J. R. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui86b] Quinlan, J. Ross. The effect of noise on concept learning. In Michalski, Ryszard S., Carbonell, Jaime G., and Mitchell, Tom M., editors, *Machine Learning. An Artificial Intelligence Approach. Volume II*, pages 149–166. Morgan Kaufmann, Los Altos, CA, 1986.

[Ren88] Rendell, Larry. Learning hard concepts. In *Proceedings of the First International Workshop on Change of Representation and Inductive Bias (Supplementary Papers)*, pages 70–100, 1991.

[RJ91] Rau, Lisa F. and Jacobs, Paul S. Creating segmented databases from free text for text retrieval. In *Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 337–346, 1991.

[RS76] Robertson, S. E. and Sparck Jones, K. Relevance weighting of search terms. *Journal of the American Society for Information Science*, pages 129–146, May-June 1976.

[Rum70] Rummel, R. J. *Applied Factor Analysis.* Northwestern University Press, Evanston, IL, 1970.

[Sal68] Salton, Gerard. *Automatic Information Organization and Retrieval.* McGraw-Hill Book Company, New York, 1968.

[Sal71] Salton, Gerard, editor. *The SMART Retrieval System: Experiments in Automatic Document Processing.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[Sal86] Salton, Gerard. Another look at automatic text-retrieval systems. *Communications of the ACM*, 29(7):648–656, July 1986.

[Sal89] Salton, Gerard, editor. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, MA, 1989.

[Sax91] Saxena, Sharad. On the effect of instance representations on generalization. In *Eighth International Workshop on Machine Learning*, pages 198–202, 1991.

[SB71] Sparck Jones, K. and Barber, E. O. What makes an automatic keyword classification effective? *Journal of the American Society for Information Science*, pages 166–175, May-June 1971.

[SB77] Sparck Jones, K. and Bates, R. G. Research on automatic indexing 1974 - 1976 (2 volumes). Technical report, Computer Laboratory. University of Cambridge, 1977.

[SB87] Salton, Gerard and Buckley, Chris. Term weighting approaches in automatic text retrieval. Technical Report 87-881, Department of Computer Science; Cornell University; Ithaca, New York, November 1987.

[SB90] Salton, Gerard and Buckley, Chris. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[Sel65] Selin, Ivan. *Detection Theory.* Princeton University Press, Princeton, NJ, 1965.

[Sem89] Sembok, Tengku Mohd Tengku. *Logical-Linguistic Model and Experiments in Document Retrieval.* PhD thesis, Department of Computing Science, University of Glasgow, August 1989.

[SFW83] Salton, G., Fox, E. A., and Wu, H. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, November 1983.

[SJ67] Sparck Jones, Karen and Jackson, David. Current approaches to classification and clump-finding at the Cambridge Language Research Unit. *The Computer Journal*, 10:29–37, 1967.

[SK86] Stanfill, Craig and Kahle, Brewster. Parallel free-text search on the Connection Machine system. *Communications of the ACM*, 29(12):1229–1239, December 1986.

[SL65] Salton, G. and Lesk, M. E. The SMART automatic document retrieval system—an illustration. *Communications of the ACM*, 8(6):391–398, June 1965.

[SL68] Salton, G. and Lesk, M. E. Computer evaluation of indexing and text processing. *Journal of the Association for Computing Machinery*, 15(1):8–36, 1968.

[SM81] Smith, Edward E. and Medin, Douglas L. *Categories and Concepts.* Harvard University Press, Cambridge, MA, 1981.

[SM83] Salton, Gerard and McGill, Michael J. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company, New York, 1983.

[Spa71] Sparck Jones, Karen. *Automatic Keyword Classification for Information Retrieval.* Archon Books, 1971.

[Spa72] Sparck Jones, Karen. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, March 1972.

[Spa73a] Sparck Jones, Karen. Collection properties influencing automatic term classification performance. *Information Storage and Retrieval*, 9:499–513, 1973.

[Spa73b] Sparck Jones, Karen. Does indexing exhaustivity matter? *Journal of the American Society for Information Science*, pages 313–316, September-October 1973.

[Spa75] Sparck Jones, Karen. A performance yardstick for test collections. *Journal of Documentation*, 31(4):266–272, December 1975.

[Spa79] Sparck Jones, K. Experiments in relevance weighting of search terms. *Information Processing and Management*, 15:133–144, 1979.

[Spa81] Sparck Jones, Karen. Retrieval system tests 1958–1978. In Sparck Jones, Karen, editor, *Information Retrieval Experiment*, chapter 12. Butterworths, London, 1981.

[SRT88] Seshu, Raj, Rendell, Larry, and Tcheng, David. Managing constructive induction using subcomponent assessment and multiple-objective optimization. In *Proceedings of the First International Workshop on Change of Representation and Inductive Bias*, pages 293–305, 1988.

[SS89] Salton, Gerard and Smith, Maria. On the application of syntactic methodologies in automatic text analysis. In *Twelfth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 137–150, 1989.

[ST84] Sparck Jones, K. and Tait, J. I. Automatic search term variant generation. *Journal of Documentation*, 40(1):50–66, March 1984.

[Sv75] Sparck Jones, K. and van Rijsbergen, C. J. Report on the need for and provision of an 'ideal' information retrieval test collection. Technical report, University Computer Laboratory; University of Cambridge, 1975.

[Sv83] Smeaton, A. F. and van Rijsbergen, C. J. The retrieval effects of query expansion on a feedback document retrieval system. *The Computer Journal*, 26(3):239–246, 1983.

[Sv88] Smeaton, A. F. and van Rijsbergen, C. J. Experiments on incorporating syntactic processing of user queries into a document retrieval strategy. In *Eleventh International Conference on Research & Development in Information Retrieval*, pages 31–51, 1988.

[Sve86] Svenonius, Elaine. Unanswered questions in the design of controlled vocabularies. *Journal of the American Society for Information Science*, 37(5):331–340, 1986.

[SW90] Sejnowski, Terrence J. and White, Halbert. Introduction to Nilsson, Nils J. *The Mathematical Foundations of Learning Machines*, Morgan Kaufmann, San Mateo, CA, 1990.

[Swe64] Swets, John A., editor. *Signal Detection and Recognition by Human Observers.* John Wiley & Sons, New York, 1964.

[Swe69] Swets, John A. Effectiveness of information retrieval methods. *American Documentation*, pages 72–89, January, 1969.

[SWW76] Salton, G., Wong, A., and Wu, C. T. Automatic indexing using term discrimination and term precision measurements. *Information Processing and Management*, 12:43–51, 1976.

[SWY75] Salton, G., Wong, A., and Yang, C. S. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, November 1975.

[SYY75] Salton, G., Yang, C. S., and Yu, C. T. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, pages 33–44, January-February 1975.

[SZB90] Salton, Gerard, Zhao, Zhongnan, and Buckley, Chris. A simple syntactic approach for the generation of indexing phrases. TR 90-1137, Department of Computer Science; Cornell University; Ithaca, New York, March 1990.

[Tag81] Tague, Jean M. The pragmatics of information retrieval experimentation. In Sparck Jones, Karen, editor, *Information Retrieval Experiment*, chapter 5. Butterworths, London, 1981.

[Tho91] Thompson, Kevin, Langley, Pat, and Iba, Wayne. Using background knowledge in concept formation. In *Eighth International Workshop on Machine Learning*, pages 554–558, 1991.

[Thu86] Thurmair, G. A common architecture for different text processing techniques in an information retrieval environment. In *Ninth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 138–143, 1986.

[TSN90] Towell, Geoffrey G., Shavlik, Jude W., and Noordewier, Michiel O. Refinement of approximate domain theories by knowledge-based neural networks. In *AAAI-90*, pages 861–866, 1990.

[Tuk57] Tukey, John W. On the comparative anatomy of transformations. *The Annals of Mathematical Statistics*, 28:602–632, 1957.

[Tuk77] Tukey, John W. *Exploratory Data Analysis.* Addison-Wesley, Reading, MA, 1977.

[Tur90] Turtle, Howard. *Inference Networks for Document Retrieval.* PhD thesis, Computer and Information Science Department, University of Massachusetts at Amherst, October 1990.

[UB90] Utgoff, Paul E. and Brodley, Carla E. An incremental method for finding multivariate splits for decision trees. In *Seventh International Conference on Machine Learning*, pages 58–65, 1990.

[Utg86] Utgoff, Paul E. Shift of bias for inductive concept learning. In Michalski, Ryszard S., Carbonell, Jaime G., and Mitchell, Tom M., editors, *Machine Learning. An Artificial Intelligence Approach. Volume II*, pages 107–148. Morgan Kaufmann, Los Altos, CA, 1986.

[Val84] Valiant, L. G. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[van72] van Rijsbergen, Cornelis Joost. *Automatic Information Structuring and Retrieval.* PhD thesis, King's College, Cambridge, July 1972.

[van77] van Rijsbergen, C. J. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 33(2):106–119, June 1977.

[van79] van Rijsbergen, C. J. *Information Retrieval.* Butterworths, London, second edition, 1979.

[van81] van Rijsbergen, Cornelis J. Retrieval effectiveness. In Sparck Jones, Karen, editor, *Information Retrieval Experiment*, chapter 3. Butterworths, London, 1981.

[VBRV87] Vickery, Alina, Brooks, Helen, Robinson, Bruce, and Vickery, Brian. A reference and referral system using expert system techniques. *Journal of Documentation*, 43(1):1–23, March 1987.

[vHP81] van Rijsbergen, C. J., Harper, D. J., and Porter, M. F. The selection of good search terms. *Information Processing and Management*, 17:77–91, 1981.

[vS73] van Rijsbergen, C. J. and Sparck Jones, K. A test for the separation of relevant and non-relevant documents in experimental retrieval collections. *Journal of Documentation*, 29(3):251–257, September 1973.

[VS87] Vleduts-Stokolov, Natasha. Concept recognition in an automatic text-processing system for the life sciences. *Journal of the American Society for Information Science*, 38:269–287, 1987.

[VV87] Vickery, Brian and Vickery, Alina. *Information Science in Theory and Practice.* Butterworths, London, 1987.

[Wil79] Willett, Peter. Document retrieval experiments using indexing vocabularies of varying size. II. Hashing, truncation, digram, and trigram encoding of index terms. *Journal of Documentation*, 35(4):296–305, December 1979.

[Wil88] Willett, Peter. Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management*, 24(5):577–598, 1988.

[WK91] Weiss, Sholom M. and Kulikowski, Casimir A. *Computer Systems That Learn.* Morgan Kaufmann, San Mateo, CA, 1991.

[Wor71] Worona, S. Query clustering in a large document space. In Salton, Gerard, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 13, pages 298–310. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[WS81] Wu, Harry and Salton, Gerard. The estimation of term relevance weights using relevance feedback. *Journal of Documentation*, 37(4):194–214, 1981.

[WV85] Wang, Yin-Chen and Vandendorpe, James. Relational thesauri in information retrieval. *Journal of the American Society for Information Science*, 36(1):15–27, 1985.

[WYB88] Wong, S. K. M., Yao, Y. Y., and Bollmann, P. Linear structure in information retrieval. In *Eleventh International Conference on Research & Development in Information Retrieval*, pages 219–232, 1988.

[YR77] Yu, Clement T. and Raghavan, Vijay V. Single-pass method for determining the semantic relationships between terms. *Journal of the American Society for Information Science*, pages 345–354, November 1977.

[YS76] Yu, C. T. and Salton, G. Precision weighting—an effective automatic indexing method. *Journal of the Association for Computing Machinery*, 23(1):76–88, 1976.

[YS77] Yu, C. T. and Salton, G. Effective information retrieval using term accuracy. *Communications of the ACM*, 20(3):135–142, March 1977.