Rapid statistical classification on the Medline database of biomedical literature

Graham L. Poulter

· July 2008 ·

Department of Molecular and Cell Biology
University of Cape Town

Submitted in fulfilment of the requirements for the degree of Master of Science

Supervised by
Prof. Cathal Seoighe

This work was supported by the Stanford-South Africa Bio-Medical Informatics Programme (SSABMI) — which is supported by US National Institutes of Health Fogarty International Center (FIC) grant D43 TW06993 — by the South African National Research Foundation (NRF), and also by the National Bioinformatics Network (NBN) and the University of Cape Town (UCT). The views and conclusions in this document are those of the author, and should not be interpreted as representing the views of any sponsoring institutions.

Abstract

Biomedical databases curate biological and medical information from the literature indexed by Medline, the world's largest bibliographic database of biomedical literature. Search interfaces such as PubMed generally suffice to retrieve biomedical journal articles, but not for curation problems, where high recall of a large body of relevant documents is required, and many different terms may indicate that a document is relevant. Knowledge engineering approaches where experts specify large sets of logical rules to detect relevant documents, are giving way to supervised learning algorithms that train a classifier of documents from a set of labelled examples. However, supervised learning also requires a great deal of effort on the part of database curators who have to develop and tune a classifier for their particular topic, construct a subset of Medline for the classifier, and label a sufficiently large training sample.

The objective of this research is to remove the aforementioned barriers to using supervised learning on Medline. The foundation is a naïve Bayes classifier using features from the title, abstract, Medical Subject Headings, journal and authors of Medline records. To remove the requirement for specifying a subset of Medline and a manually-labelled training sample, the classifier uses unlabelled documents in Medline to approximate irrelevant training data. This prepares the classifier to rank all of the nearly 17 million documents in Medline, and requires only readily-available relevant training examples as input. However, the usual method of estimating naïve Bayes parameters biases them towards the least prevalent class, and we introduce a novel bias correction that greatly improves performance when relevant documents are rare. To mitigate the need for developing and tuning a new classifier for each problem, we optimise the formulation not for a particular topic but for Medline records in general by tuning the formulation on four test corpora of widely varying characteristics, and make the classifier re-usable by providing an on-line service. To achieve the latter we developed pre-processing steps that reduce the classification phase to 168 seconds, whereas implementations without this pre-processing would take several days.

In testing under cross-validation, the classifier matches the performance of an existing specialised database curation classifier on its own data. On large samples of Medline, it detects about two thirds of a set of relevant records detected by a knowledge-engineering query for the same number of results, when trained with older knowledge-engineering results, and has high precision in the upper ranks. The classifier is intended for databases lacking a supervised-learning curation filter, but can augment existing filters, and the simplicity of the service makes it useful for extending any topical collection of Medline citations, such as a comprehensive bibliography on a topic of interest to an individual researcher.

Contents

List of Figures

List of Tables

Acknowledgements

I thank Prof. Cathal Seoighe for supervising the research. He was supportive throughout the project, reviewed dozens of drafts of the manuscript for BMC Bioinformatics, and provided expert advice on the art of scientific writing in addition to consultations on the research itself. He also provided funding through the National Bioinformatics Network (NBN) to present a poster on the work at the 2007 ISMB/ECCB conference and to publish in BMC Bioinformatics. I thank the remaining co-authors on the BMC Bioinformatics manuscript, Prof. Russ B. Altman and Dr Daniel L. Rubin of Stanford University, for their support and insight. In the early days of the project we would confer via email and teleconferencing to brainstorm solutions to problems I was encountering. I also thank Tina Zhou for administering the server on which http://mscanner.stanford.edu, the on-line service resulting from the research, is hosted. I thank the members of the executive committee of the Stanford-South Africa Bio-Medical Informatics (SSABMI) programme over the course of my degree: Prof. Russ B. Altman, Prof. Win Hide, Prof. Cathal Seoighe, Dr Betty Cheng, Patricia Josias and Judith Jansen for their all their kind support and assistance. The members of the UCT Computational Biology research group have always been good company, including Natasha Wood, Andre Faure, Ryan Goosen, Renaud Gaujoux and Joël Ravelomanantsoa-Ratsimihah in our room; Dr Konrad Scheffler, Dr Wayne Delport, Venu Vuppu, Halima Rabiu, Bukiwe Lupindo, Victoria Nembaware, Nobubelo Ngandu and Prof. Nicola Mulder in the other; and Rodger Duffett whose technical skills are responsible for the lab's excellent computing infrastructure. I thank my mother and father for their support and encouragement in our weekly phone calls.

Cape Town, July 2008

Chapter 1

Introduction

Medline, maintained by the U.S. National Library of Medicine (NLM), is the world's largest bibliographic database of biological and medical literature (NLM 2007d). PubMed (NLM 2006a) is the NLM's official search interface to Medline. It uses pure boolean queries, returning all records that satisfy the conditions expressed by the query. Although boolean queries are powerful, a substantial amount of research has gone into developing strategies for retrieving biomedical literature that are more effective in certain scenarios. For example, Relemed (Siadaty et al. 2007) provides a form of relevance ranking, EBIMed (Rebholz-Schuhmann et al. 2007) clusters results according to biological named entities extracted from the text, and the PubMed related articles service (Lin and Wilbur 2007) makes it easier to explore topic space by retrieving articles similar to a given article of interest. There are, however, scenarios where ad hoc information retrieval on Medline has generally proven unsuitable, in particular for the task of identifying Medline records relevant to databases that are curated with the aid of information extracted from biomedical literature. This task requires high recall since curators want to review all relevant documents, but retrieval is hindered by the large number of terms which could indicate that an article is relevant to the database.

To retrieve Medline records representing articles of relevance to them, a growing number of biomedical databases have implemented strategies based on text classification using supervised learning. Classification consists of placing objects into categories, and the task of retrieving Medline records can be formulated as a problem of placing each Medline record

into exactly one of the two classes: relevant and irrelevant. A classifier of Medline records can be constructed manually by specifying a boolean query for each category of interest, also known as knowledge engineering (Sebastiani 2002). This requires a great deal of work on the part of domain experts, yet has low precision in the database curation scenario (Wang et al. 2007). Supervised learning for classification instead uses a machine learning algorithm that inductively constructs a classifier from a set of manually classified examples. Supervised learning effectively handles the case of many relevant features (Sebastiani 2002, Joachims 1998), and can achieve higher recall and precision more easily than manually specifying combinations of features that indicate a relevant article. Supervised learning has been used by the Pharmacogenetics Knowledgebase (PharmGKB) (Hewett et al. 2002), Textpresso (Müller et al. 2004), the Immune Epitope Database (IEDB) (Wang et al. 2007), the Biomolecular Interaction Network Database (Donaldson et al. 2003) and several others discussed in Section 2.5. Instead of Medline records, the 2005 TREC Genomics Track (Hersh et al. 2005) set the problem of classifying full-text articles using supervised learning, to identify articles for Mouse Genome Database curation (Eppig et al. 2005). The use of PubMed related articles has also been suggested for updating a bibliography (Liu and Altman 1998) or a database (Perez-Iratxeta et al. 2003).

If using supervised learning to locate relevant Medline records were easy, it would be useful not only in database curation, but in other scenarios where relevant training examples are available — such as maintenance of a comprehensive bibliography on some topic, or the construction of a corpus for text mining. The first barrier to supervised classification of Medline is the implementation of a classifier. It is is still a lot of work adapt a machine learning algorithm to work with a new data type, although in simple cases it is less work than constructing a complex classification rule for a given topic using knowledge engineering. There exist general-purpose machine learning programs such as WEKA (Frank et al. 2004), but these still require background in text classification to create an effective classifier, and a custom preprocessing step to extract features from text. WEKA also becomes slow on large data sets, as the IEDB researchers found (Wang et al. 2007). A second barrier to supervised Medline classification is the construction of training data, which typically requires defining a subset of Medline, manually classifying all or part of it, and applying the trained classifier to future members of the subset. The IEDB already has a manually classified subset from their earlier curation filter based on boolean queries, but often only relevant examples are initially available, as is the case for the PharmGKB database (Hewett et al. 2002) and non-database scenarios.

1.1 Objectives of the research

This research aims to lower barriers to identifying relevant Medline records using supervised learning, while retaining acceptable classification performance. In the biomedical database scenario, classifiers have historically been specialised for a particular database. To overcome that limitation, our first goal is to make a classifier that is effective over a range of topics and input sizes yet specialised and optimised for Medline records in general. If supervised classification of Medline is to be applied in scenarios other than identifying literature for biomedical databases, it should not require a local Medline repository, or a great deal of effort to prepare the classifier for use. Our second objective is therefore to make the classifier available in the familiar form of a Medline retrieval service — even though the input will be training examples instead of the queries used by ad hoc information retrieval.

The third objective is to perform the classification on all of Medline instead of a subset of Medline. Medline subsets typically contain thousands of articles, and are constructed using a pre-filter of boolean conditions to capture most relevant articles, thus reducing the number of documents the classifier has to work with. However, the retrieval problems which benefit most from supervised learning are those which make it most difficult to capture most relevant articles with a simple filter. The IEDB classifier (Wang et al. 2007) operated on a Medline subset, and the filter query was not trivial. Requiring knowledge engineering to make a subset for supervised learning negates the main advantage of supervised learning: that it much easier to apply to a new problem than knowledge engineering. Operating on all of Medline, and providing an on-line retrieval service in turn require a fourth objective: making the classifier exceptionally fast. Maximum efficiency requires a simple Bayesian classifier (McCallum and Nigam 1998), specialised for Medline records, and highly optimised data structures. Using all of Medline for supervised learning instead of a subset has also been considered before: it was used for by the PharmGKB (Rubin et al. 2005), suggested but not implemented by Suomela and Andrade (2005), and partially implemented in the defunct PubFinder service (Goetz and von der Lieth 2005) – discussed further in Section 2.5.

Classifiers ideally use "gold standard" training sets, where a random sample is drawn from the domain of operation (such as a subset of Medline), and manually classified to produce an initial corpus for training and testing. In addition to constructing the subset, manually classifying a large sample adds to the effort required to use supervised learning.

Finally, with the whole of Medline as a domain, the prevalence of documents relevant to a given topic is so low that only a handful will appear in a sample of thousands. Fortunately, biomedical databases and other sources often provide a ready-made set of *relevant* documents. Our objective of making the classifier easy to apply means developing a method that only requires the relevant examples for input, and completes training by using unlabelled documents from the rest of Medline. We do this by treating rest-of-Medline as approximately irrelevant, because probabilistic methods are robust to pollution of the irrelevant training data by a low prevalence of relevant documents.

In summary, the objective of the research is a supervised learning classifier that is optimised for peculiarities of Medline records but not for a particular subject domain, presents as an on-line retrieval service, classifies the whole of Medline (no need to construct a subset), does so quickly enough for on-line use, only needs relevant examples from users, and all the while maintains comparable performance to existing special-purpose classifiers on their own data. Satisfying the objective contributes toward the goal of lowering the barrier for databases to begin using supervised learning, and makes it feasible to apply supervised Medline classification in other scenarios. The methodology is to iteratively test the method against the stated objectives and devise improvements where necessary. The tests of the performance and efficiency objectives are are described in Section 3.4, and the revisions of the method discussed in Section 5.2. The activities involved in developing the method can be broken down as follows:

1. Gather information about the problem then design and implement a method of using supervised learning to classify all of Medline, featuring a Web front-end that requires only relevant examples to be specified.

2. Iteratively improve the performance of the method by testing it under cross validation on a wide range of topics and input sizes and revising the method for better performance across the full range of cases. Then test the performance objective by comparing the resulting method to an existing classifier on its own specialist topic.

3. Iteratively improve the efficiency of the method by identifying steps that dominate the execution time and implementing techniques to make them faster.

1.2 Road map

Chapter 2 covers the different information retrieval strategies currently used with Medline. It provides background on supervised learning in text classification, to locate the problem of classifying Medline for relevant documents in the broader context of text classification problems, algorithms for constructing classifiers, and evaluation of classifier effectiveness. It then examines the information available in Medline records and biomedical text and methods of representing it for learning algorithms. Lastly, it examines related research on Medline classification, focusing on supervised learning approaches and the task of database curation.

Chapter 3 first derives a naïve Bayes classifier for binary classification of Medline records. It presents different methods for smoothing the estimates of classifier parameters, including a new variation on Laplace smoothing for extremely-skewed training data that was introduced in response to poor performance of the method under high class skew. It then presents methods for extracting features from Medline records, reducing the dimensionality of the feature space (also introducing an improvement for skewed training data), and how we optimised classifier speed with special data structures. It then explains how the component methods work together to filter Medline, and the design of the web application. Lastly, it describes the initial corpora for cross-validation, and the design of cross-validation experiments for optimising the classifier formulation and evaluating its effectiveness.

Chapter 4 presents results of cross-validation experiments to determine the best methods for smoothing parameter estimates, selecting features, and indexing Medline records, avoiding overfitting to one subject by using several data sets with radically different properties. It then looks more deeply at the final classifier, with experiments to elucidate why it performs as it does on the different corpora. It then compares the final classifier to the earlier version presented in Poulter et al. (2008), compares it to an independent classifier by Wang et al. (2007), and evaluates classifier performance on a test that mimics conditions in operation. Lastly, it examines the web interface through an example application of the classifier and measures the speed of classification.

Chapter 5 first critically evaluates the effectiveness of the classifier, then weighs up the design of the classifier and evaluation experiments against alternatives. It then discusses the efficiency of the classifier, and its practical applications. Lastly, it suggests ways in

which the classifier might be improved or extended.

In Chapter 6 we draw conclusions from the discussion, in light of the objectives set out in this chapter.

Chapter 2

Background

This chapter examines related work and background in the problem of filtering Medline records. We discuss services for information retrieval on Medline, then distinguish knowledge engineering (expert PubMed queries) and supervised learning. We discuss the underlying techniques needed to perform supervised learning on Medline: the families of classifier algorithms (focusing on naïve Bayes classifiers), influences on classifier performance, and the caveats of evaluating classifiers in Medline filtering scenarios. There are also unique opportunities and caveats in extracting and selecting features for classification from the structured information found in Medline records. Within the context of Medline filtering, we look at particular Medline filters, mainly for database curation, and finish with the problem of filtering all of Medline instead of a subset.

2.1 Retrieval of biomedical literature

Medline (NLM 2007d) is the worlds largest bibliographic database of biomedical literature, containing abstracts and citation information on 16,880,015 articles in biology and medicine at the beginning of 2008 as reflected in the Baseline distribution of Medline (NLM 2007b). Medline indexes about 5,000 journals, with coverage beginning primarily from 1950. Its scope is biomedicine and health, broadly interpreted to include most life sciences research and relevant sub-fields of engineering and chemistry. Medline is also growing rapidly: at time of writing PubMed indicates that 740,826 Medline records were entered in 2007: an

increase over the 700,994 records entered in 2006, and equating to 2,029 publications for each day in the year.

Below we describe services for retrieving literature from Medline that work from a query expressing a topic or information need. These are in contrast to the supervised text classifiers of Section 2.5 that infer a classifier of documents from labelled examples then filter Medline for relevant literature. The two approaches do, however, share underlying methods for representing text (Section 2.4) and evaluating performance (Section 2.3). It is also possible to incorporate supervised learning into information retrieval, using machine learning to refine the result set based on user feedback about which results were relevant. On the other hand, Chapter 3 uses pure supervised learning, but imitates information retrieval by using ranked classifier outputs and hosting the classifier behind a Web interface.

2.1.1 Boolean queries

PubMed (NLM 2006a) searches Medline (and some additional records), and uses structured boolean queries that specify the conditions that a record must satisfy to be included in the results. The results are reverse-ordered by date of publication. A boolean query is constructed using AND, OR and NOT operators to combine limits on the values that may be taken on by record fields such title, abstract, author, journal, publication date and Medical Subject Headings (NLM 2007e). MeSH is a controlled vocabulary of terms used to indicate major topics and other subject-defining attributes of a Medline record. The boolean formalism is powerful — capable of retrieving any subset of the database, and forms the basis of the knowledge engineering approach to building a classifier as discussed in Section 2.2.1. However, boolean queries are not always the best tool for a particular retrieval job. Hersh and Hickam (1998), for example, found that physicians who were also novice Medline searchers were no more successful using the advance the features of boolean queries than when simply entering text words, and that in either case only a fraction of the literature relevant to their needs is actually retrieved. In recent years, a research theme has developed around strategies for retrieving biological and medical literature more effectively in particular scenarios.

2.1.2 Relevance ranking

A major problem when searching with boolean queries is that, except for the most specific of queries, the result set is large and contains a low proportion of relevant results. One way to mitigate the problem is to rank results by their relevance to the query, so that the most relevant articles are concentrated at the top. Casual searchers typically need just a few documents, and ranking by relevance greatly eases the task. HubMed (Eaton 2006), for example, can re-order PubMed results according to the frequency of query terms in the records, and has other enhancements such as clustering results, recommending related query terms, and providing links to definitions of terms, abbreviations and gene names. The Relemed (Siadaty et al. 2007) search engine evaluates a boolean expression on individual sentences from the title and abstract of each record, and MeSH terms too. The highest ranked results have a match in all three fields, and in the lowest-ranked results no individual sentences match the query text but the full text of the record does. Another retrieval service, botXminer (Mudunuri et al. 2006), uses the search capabilities of Oracle XML databases to search the XML distribution of Medline. These services can greatly speed up the process of finding relevant literature for individual researchers.

A number of services are based on the cosine similarity vector space model (Salton 1989), which ranks documents by their similarity to the query as indicated by the angle between each document vector and the query vector. Google Scholar (Google 2008), Healthline (Healthline 2008) and Medscape (Medscape 2008) are retrieval services that base their ranking at least partly on vector space retrieval in addition to boolean criteria. These services do not index Medline exclusively; Healthline and Medscape search health literature from several resources, and Google Scholar indexes scientific literature in general, including much of the biomedical literature included in Medline. More recently Mao and Chu (2007) have developed a phrase-based vector space model for Medline retrieval. Keyword results can also be ranked using probabilistic retrieval, which is related to Bayes classification and covered in Section 2.2.3.

2.1.3 Related article search

Related article search, exemplified by the PubMed related articles feature (Lin and Wilbur 2007), retrieves the documents most closely related to a given document of interest. It

helps when exploring the literature, as a more effective alternative to searching for keywords found in interesting results. The method used by PubMed (Lin and Wilbur 2007) calculates the similarity between a pair of documents using a probabilistic model over all topics (approximated by vocabulary terms) to evaluate the probability that a reader would be interested in one article given interest in the other. As this is a computationally expensive process, PubMed pre-computes the closest neighbours of each Medline record so they can be looked up instantly. As we see in Section 2.5.4, related articles can also be used in database curation.

A simple way to search for related articles is to use the text of the interesting abstract as a query with an existing information retrieval method. Lin and Wilbur (2007) in fact evaluate PubMed Related Articles by re-ranking the results of using the abstract as a query under the Okapi BM25 probabilistic retrieval model (Robertson et al. 1992). Tbahriti et al. (2005) likewise use the text of the abstract as a query in ordinary vector space retrieval (Salton 1989), and propose an independent measure of relatedness using the overlap between the citation lists of a pair of documents. Related article search is not possible in ordinary search engines, as they are designed to accept a few key words as input. A text similarity search like eTBlast (Lewis et al. 2006) however uses an arbitrary passage of text for input, and takes a few minutes to rank Medline abstracts by their similarity to the input. The more recent JANE tool (Schuemie and Kors 2008) also takes abstracts as input, but is much faster because it uses a standard vector-space model from the efficient Lucene search library (Gospodnetic and Hatcher 2005). Results are grouped by author and journal, as it is designed to assist with finding appropriate journals and potential reviewers for a manuscript. On an abstract from a published Medline record, JANE results are similar to the PubMed related articles for that record.

2.1.4 Extraction of biological and medical named entities

Biomedical literature retrieval can be enhanced by deliberately making use of biological and medical named entities — such as genes/proteins, drugs, species, diseases — and the controlled vocabularies, such as Gene Ontology (Consortium 2008), that organise those entities. This takes biomedical information retrieval into the domain of text mining, in particular information extraction and relationship detection for biomedical named entities. The simplest approach to extracting named entities is a dictionary, and co-occurrence

of entities in the same sentence is a simple way to infer relationships between entities. The EBIMed search engine (Rebholz-Schuhmann et al. 2007), for example, extracts such named entities, maps them onto Gene Ontology terms, and groups results accordingly. GoPubMed (Doms and Schroeder 2005) also clusters results according to inferred Gene Ontology terms. More advanced methods for relationship extraction use Natural Language Processing (NLP) to interpret text. Chilibot (Chen and Sharp 2004), for example, uses NLP to construct relationship networks amongst named entities found in results. Another resource specific to scientific literature is figures and captions. The BioText (Hearst et al. 2007) search engine uses these, searching within captions to display relevant figures when searching Open Access papers from the NLM's PubMed Central repository (PMC 2008).

2.2 Background on text classification

Text classification is the process of placing documents into categories. It has historically been performed using knowledge engineering to manually construct classification rules, but is being superseded by supervised learning to automatically construct a classifier by learning from labelled examples. To provide context for the Medline filtering methods in Section 2.5, we also discuss the families of supervised learning algorithms that are used in text classification.

2.2.1 The knowledge engineering approach

Knowledge engineering was the original approach to text classification, in which experts construct logical rules for each category of interest. Such a classifier is also known as an "expert system". A document is classified under a given category if it matches the characteristics specified in the classification rule. Knowledge engineering has also been extensively applied to Medline records, where the logical conditions are expressed in the form of a boolean PubMed query. The Immune Epitope Database (Peters et al. 2005a, Wang et al. 2007) originally used knowledge engineering to filter Medline using a large PubMed query composed of many smaller queries, one of which is shown in Figure 2.1. The results were manually examined and relevant articles forwarded to the database curators. Although the IEDB used several queries, the result is equivalent to a large query in

```
(epitope[TW] OR epitopes[TW] OR mimotope[TW] OR ((MHC[tw] OR "major
histocompatibility complex"[tw] OR HLA[tw]) AND (peptide[tw] OR
peptides[tw])) OR "TCR recognition"[tw] OR ("Class"[tw] AND "I motif"[tw])
OR supermotif[tw] OR immunogenic linear OR ("peptide-based"[tw] AND
CTL[tw]) OR phage displa*[tw] OR "antibody binding"[tw] OR "protective
immune response"[tw] OR antibody recog*[tw] OR "cytotoxicity assay"[tw]
OR "new monoclonal"[tw] OR "novel antibody"[tw] OR ( (monoclonal
antibod*[tw]) AND "binding site"[tw]) OR ( (KA[tw] OR KD[tw]) AND
(monoclonal[tw] OR mAb[tw])) OR "neutralizing antibody"[tw] OR "peptide
vaccine"[tw] OR (peptide conjugate vaccine*[tw]) OR ((CD8[tw] OR CD4[tw])
AND "T cells"[tw] AND (peptide[tw] OR peptides[tw])) OR ("antigenic
repertoire"[tw]) OR ((peptide[tw] OR peptides[tw]) AND "antibody
reactivity"[tw]) OR ("Class II"[tw] AND (binding [tw] OR bound[tw]
OR peptide[tw] OR peptides[tw])) OR "immunogenic peptide"[tw]) AND
("HIV"[Text Word] OR ("AIDS"[Text Word] AND "virus"[Text Word]) OR
"Human immunodeficiency virus"[Text Word]) AND (hasabstract[text] AND
English[Lang] AND ("1900"[PDAT] : "2005/12/31"[PDAT])) NOT (Review[PT] OR
Editorial[PT] OR meta-Analysis[PT] OR Comment[PT])
```

Figure 2.1: A query fragment used by the Immune Epitope Database (IEDB) (Wang et al. 2007) in the knowledge engineering phase of its curation filter. The IEDB filter rule is divided into many PubMed queries and this is one of smaller query fragments, which detects IEDB-relevant publications relating to HIV/AIDS. Each query returns a constant set of results because of [PDAT] limits on publication date.

which the components are joined using the "OR" operator. Knowledge engineering is also used by the NLM for the PubMed subset strategies (NLM 2006b). A PubMed subset may contain millions of records and the strategy for constructing it is a PubMed query containing hundreds or thousands of conditions. A PubMed searcher may target their search to a subset such as "AIDS" (NLM 2007a) or "bioethics" (NLM 2007c). Knowledge engineering also has a long history in identifying clinical evidence articles. Wilczynski et al. (2005) describe the design and construction of a Medline search strategy for clinical studies of health disorders, important for evidence-based medicine and systematic reviews. The main drawback of knowledge engineering or expert systems is the knowledge acquisition bottleneck (Sebastiani 2002). Updating a classification rule requires intervention from experts in the subject domain and experts in knowledge engineering, whenever the requirements of the filter or the characteristics of relevant documents change. Also, the exercise is repeated for each new classifier: effort put into creating a good classifier for one problem does not carry over to other subject domains.

2.2.2 The supervised learning approach

In supervised learning, an algorithm inductively constructs a classifier from manually labelled examples, and the classifier in turn assigns categories to unseen instances. Supervised learning eases the knowledge acquisition problem: even if training examples are not already available it is easier for experts to manually classify examples than to express as a logical formula all the criteria being used to make those judgements. The learning algorithms can also be adapted to new domains, even as they equal or exceed the performance of the best expert systems (Sebastiani 2002).

Supervised classification is formally a problem of function approximation. One can imagine a target function $\breve{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ that perfectly assigns a true or false value to each point in the space of documents $\mathcal{D} = \{d_1, \ldots, d_N\}$ and categories $\mathcal{C} = \{c_1, \ldots, c_k\}$ (Sebastiani 2002). The true/false output indicates whether each document is or is not a member of a given category. The learning algorithm is given training documents that have been manually classified (to approximate the operation of $\breve{\Phi}$), and inductively constructs a classification rule $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ that predicts the class membership of any given document. The effectiveness of the classifier Φ at approximating $\breve{\Phi}$ is measured on a set of test documents by comparing its predicted classes to manually assigned classes — where no knowledge about the test documents was available to the learning algorithm (see Section 2.3).

There are distinct sub-problems within document classification. In multi-label classification or overlapping categories, each document is to be placed into zero or more categories. The Medical Text Indexer (Aronson et al. 2004, Gay et al. 2005), for example, performs multi-label classification by suggesting Medical Subject Headings for Medline curators to assign to incoming Medline records as part of the Medline Indexing Initiative (Aronson et al. 2000). In single-label classification or non-overlapping categories, each document must be placed in precisely one category. A special case of single-label classification is binary classification, in which there are two complementary categories: "relevant" and "irrelevant" to a given topic. The relevant/irrelevant distinction is artificial, but classifiers can rank documents to implicitly account for degrees of relevance. Filtering Medline for relevant documents can therefore be framed as a binary classification problem. Binary classifiers can in turn be used to perform multi-label classification by transforming the multi-label problem into independent binary problems to determine whether a document belongs in each category.

An additional distinction can be made between document- and category-pivoted classifiers. A document-pivoted classifier examines each document in turn and chooses categories, as occurs when indexing new Medline records with MeSH terms. A category-pivoted classifier examines each category in turn and chooses documents. Medline filtering is then the category-pivoted evaluation of documents for membership in the relevant class.

Many classifiers rank their results instead of making "hard" classification judgements like a boolean query. The category-ranking (document-pivoted) classifiers list categories in decreasing order of preference while document-ranking (category-pivoted) classifiers rank documents by order of preference for a given category. Whereas hard classifiers directly approximate $\breve{\Phi}$, ranking classifiers assign a score for each document-category pair: $S : \mathcal{D} \times \mathcal{C} \rightarrow \mathbb{R}$. Selection of a decision threshold τ produces hard judgements with the rule $\Phi(d, c) = \mathrm{T} \iff S(d, c) > \tau$. The threshold is a model parameter which may be chosen on a theoretical basis, optimised, or chosen implicitly by selecting some number or proportion of the top ranking categories or documents. Classifiers for identifying Medline records relevant to a particular topic use document ranking for the category "relevant", setting a cut-off score τ or a maximum number of documents to return.

For completeness we mention unsupervised learning, or clustering, which is used for classification when the specific categories are not defined in advance. Clustering algorithms group documents together based on their characteristics, and may also suggest a class label for each of the groups. Clustering has been used to group results in some of the retrieval systems in Section 2.1, and clusters of terms have been used for dimensionality reduction as discussed in Section 2.4.3. Clustering has also been used to organise literature in the Textpresso resource (Chen et al. 2006).

2.2.3 Learning algorithms used in text classification

Learning algorithms and classification rules may themselves be categorised. What follows is a conceptual overview of the major families of supervised learning algorithms in text classification, all of which have been used in the Medline filtering methods described Section 2.5. We examine the naïve Bayes family of classifiers and their statistical foundation in greater detail, since one of these is at the core of our method described in Chapter 3.

Linear classifiers

Many supervised learning algorithms represent each document in a collection as a feature vector $\mathbf{f} = (f_1, \ldots, f_{|\mathcal{V}|})$. $|\mathcal{V}|$ is then the number of terms in the vocabulary or, more generally, the number of features in the feature space, and f_i indicates the importance of the ith vocabulary term to the document. For linear classifiers, the score of a document for a particular category reduces to the dot-product of a weight vector $\mathbf{w}$ of term weights (for that category) and the feature vector $\mathbf{f}$, plus a constant b (Lewis et al. 1996):

$$S(\mathbf{f}, \mathbf{w}) = \mathbf{w} \cdot \mathbf{f} + b = \sum_{i=1}^{|\mathcal{V}|} w_i f_i + b \tag{2.1}$$

Learning algorithms that produce linear classifiers are distinguished by the way they calculate the weight vector $\mathbf{w}$. The Rocchio classifier, for example, is a linear binary classifier derived from the Rocchio method of information retrieval with relevance feedback (Rocchio 1971). Feature vector elements are set to term frequency times inverse document frequency (TF.IDF) (Salton 1989), meaning that the importance of a term to a document is proportional to the number of times it occurs in the document, and inversely proportional to the total number of documents in which the term appears. The Rocchio weight vector scores each document by its distance from the centroids of the relevant versus irrelevant training vectors. It differs from the standard vector space model of information retrieval, in which the retrieval score is the dot product of normalised TF.IDF vectors representing the query string and the document (Lewis et al. 1996).

The perceptron learning algorithm also produces a linear classifier, but is modelled on an artificial neural network (ANN) that iteratively updates the weight vector in response to incorrect classifications (Dagan et al. 1997). Also, support vector machines (SVMs) have recently been adapted to text categorisation (Joachims 1998). SVMs represent each training document as a point $(\mathbf{f}, y)$ in a $|\mathcal{V}| + 1$-dimensional space, where $y \in \{+1, -1\}$ means the document is or is not a member of the category. The weight vector $\mathbf{w}$ is chosen using the principle of structural risk minimisation, which involves solving the optimisation problem of finding the hyperplane $\mathbf{w} \cdot \mathbf{f} - b = 0$ having the maximum Euclidian distance to the closest training examples (Joachims 2001; page 1). ANNs and SVMs can be made non-linear, the former by inserting a layer of artificial neurons having a non-linear transfer

function, and the latter by using a more general kernel function in place of a dot product to transform the feature space.

Probabilistic classifiers

Probabilistic classifiers predict the most probable class of a document, under certain assumptions about the distributions that generate the documents and the estimated parameters of those distributions. The guiding principle is to choose the most probable class given the data — the maximum a posteriori (MAP) hypothesis — using Bayes' rule to evaluate the posterior probabilities of each class (given the document) in terms of probability of the document under each class, and the prior probabilities of the class and document:

$$c_{\text{MAP}} = \underset{c \in \mathcal{C}}{\operatorname{argmax}} P(c|d) = \underset{c \in \mathcal{C}}{\operatorname{argmax}} \frac{P(c)P(d|c)}{P(d)} \tag{2.2}$$

Where $\mathcal{C}$ is the set of categories and $P(d|c)$ is short for $P(D = d|C = c)$, where the lowercase letters d and c are the particular document and category taken on by the random variables D and C. The frequency of class c in the training data estimates the class' prior probability $P(c)$, and $P(d)$ (prior probability of the document) is often discarded because it does not affect c_{MAP}. In order to calculate $P(d|c)$, the methods assume that documents are generated by a mixture with one component distribution for each class. To generate a document, one would first select a class c with prior probability $P(c)$, and then generate a document using the corresponding mixture component (McCallum and Nigam 1998). The learning algorithm trains the classifier by estimating parameters θ_c for each mixture component, so that the classifier can calculate $P(d|c; \theta_c)$. The MAP classifiers we discuss produce just one estimate of θ, which contains all the parameters of the model, and is known as the hypothesis. In contrast to a MAP classifier, a "Bayes optimal" or "Bayesian" classifier would average its prediction over all possible hypotheses, with each hypothesis weighted by its posterior probability given the training data (Mitchell 1997; section 6.7).

Representing the document d with a feature vector $\mathbf{f}$, the naïve or simple Bayes family of MAP classifiers use "naïve" distributions that assume conditional independence amongst the elements of the feature vector with respect to the class variable, so that the joint

probability of the document features can be written as a product over individual feature probabilities:

$$P(d|c) = P(\mathbf{f}|c) = P(f_1, f_2, \ldots, f_{|\mathcal{V}|}|c) = \prod_{i=0}^{|\mathcal{V}|} P(f_i|c) \tag{2.3}$$

The resulting classifier is log-linear because the logarithm of $P(c|d)$ then reduces to a sum over the elements of the document vector. This structure is shared by all Naïve Bayes classifiers, which then differ in the statistical model used to obtain $P(f_i|c)$. The multivariate Bernoulli model assumes that documents are generated by a series of $|\mathcal{V}|$ coin-flip Bernoulli trials. Each f_i in the feature vector $\mathbf{f}$ is then a 0/1 binary value for the absence/presence of term i in the document. In Equation 2.3, the probability $P(F_i = 1|C = c)$ is the Bernoulli parameter θ_{ci}, and the probability of non-occurrence $P(F_i = 0|C = c)$ is $1 - \theta_{ci}$. The learning algorithm estimates each parameter θ_{ci} from the fraction of class c training documents that have term i. In Section 3.1.1 of the methods we formulate multivariate naïve Bayes for two classes (relevant/irrelevant) so that it ranks documents by their odds of relevance. This is known as the binary independence model (BIM) (Lewis 1998), and is related to the binary independent retrieval method of probabilistic information retrieval, which ranks documents by odds of relevance to terms in a query string and updates the Bernoulli parameters from relevance feedback (Fuhr 1992).

To take into account that words occurring more times in a document are more important to the document, we can instead define f_i to be the term frequency, which is the number of times term i occurs in the document. This leads to multinomial naïve Bayes, which models documents on a multinomial distribution with $|\mathcal{V}|$ parameters. f_i is then number of successes for observing word i in a number of independent trials equal to the length of the document (McCallum and Nigam 1998, Nigam 2001). The learning algorithm estimates the ith parameter of the multinomial distribution for class c from the fraction of the total term occurrences in class c documents constituted by term i. The multinomial classifier is also related to the logistic regression (LR) / maximum entropy method (MEM) classifier, which predicts probabilities without assuming a particular distribution. Juan et al. (2007) show that the LR/MEM classifier is equivalent to multinomial naïve Bayes using parameters that have maximum likelihood under the conditional distribution of class given document — instead of the usual parameters which can be shown to have maximum likelihood under

the joint distribution of document and class. Other models of word occurrences include Poisson distributions (Lewis 1998), which assume that different words are independent but that occurrences of the same word are spread evenly across the text. One approach using multivariate Poisson reduces to the same log odds of class membership as multinomial in a special case (Kim et al. 2006).

Decision tree and example-based classifiers

Linear and log-linear classification rules divide up the $|\mathcal{V}|$-dimensional space of possible documents in a simple manner that leaves a range of decision rules unrepresentable — for example, the rarely encountered "exclusive or" rule that judges a document relevant if either of two features are present but not if both are present. Other classifiers, such as decision trees (Mitchell 1997; Chapter 3), can express the entire space of classification rules. With a trained decision tree, the classifier evaluates conditions on document features, starting from the root of the tree and recursively taking the appropriate branch until reaching a leaf node that predicts the class of the document. Another method, which can model disjoint regions of class membership, is the k-Nearest Neighbours (kNN) classifier (Sebastiani 2002, Yang and Liu 1999). Unlike the methods discussed so far, example-based methods like kNN do not use a learning algorithm to construct a classification rule but classify unseen documents by comparing them to the training examples. The kNN method evaluates the distance between the test document and all training documents, and chooses the most popular class amongst the closest k training documents, weighting the "vote" of each training document by its distance. Methods have been developed to combine kNN with naïve Bayes to improve the latter's ranking of test documents (Jiang and Zhang 2005).

2.2.4 Factors influencing naïve Bayes performance

Empirical performance of naïve Bayes classifiers in the literature is mixed. Yang and Liu (1999) found that in a category-ranking multi-class problem on news articles, multinomial naïve Bayes underperformed against several other classifiers when the number of training examples was low, but all methods performed similarly when many training examples were available for a class. In classification on biomedical abstracts, Donaldson et al. (2003) and Aphinyanaphongs et al. (2005) found that an SVM outperformed a naïve Bayes classifier,

Wang et al. (2007) found a naïve Bayes classifier outperforming an SVM, a neural network and a decision tree, and Rubin et al. (2005) found that logistic regression outperformed a naïve Bayes classifier when the feature space was limited. Across different studies, performance is influenced by the classifier family, the particular member of the classifier family, the method of parameter estimation, the characteristics of the training documents, the experimental design and choice of performance statistic (Section 2.3), and the document representation: especially the way in which features are extracted from text (Section 2.4.1) and the features that are selected (Section 2.4.3). Any empirical comparison between classifier families is likely to make choices in the other influences that may or may not be optimal for a given classifier.

On a theoretical footing, naïve Bayes classifiers are optimal for a given document representation when the document feature vectors follow the assumptions of the statistical model. All naïve Bayes classifiers make independence assumptions as in Equation 2.3 that are violated in real-world data (Lewis 1998). For example, observing "mouse" in an abstract raises the probability that the abstract also contains the word "liver", which it would not if the occurrences were independent. In binary classification naïve Bayes may really be making use of a weaker "linked dependence" assumption specific to each document, which states that the departure from independence in the document is by a constant factor that is the same in relevant and irrelevant documents (Cooper 1995). Unaccounted-for dependencies make the estimated probabilities more extreme, and so also affects ranking. That is, if relevant documents were manually assigned degrees of relevance, naïve Bayes would disorder the optimal ranking to some extent. However, classification only depends on correctly identifying the class of greatest probability, which makes naïve Bayes judgements optimal under a wider range of conditions (Domingos and Pazzani 1997). Methods of incorporating term dependencies include learning a Bayes net classifier (Friedman et al. 1997) (naïve Bayes being a special case of a network with no dependencies), and the use of Markov random fields to represent dependencies (Metzler and Croft 2005).

The different document models also influence naïve Bayes performance. Multivariate Bernoulli only models term absence/presence and not the number of times a term occurs, which becomes increasingly important in longer texts. Multinomial outperforms multivariate Bernoulli on plain text (McCallum and Nigam 1998), and in turn was recently outperformed by a multivariate Poisson model with document length normalisation (Kim et al. 2006), although this has not been the case in the past (Lewis 1998). Parameter esti-

mation in multinomial is unduly influenced by long documents, which contribute most to the word frequencies (Lewis 1998), leading length normalisations to be proposed (Rennie et al. 2003). The multinomial model additionally assumes that occurrences of the same word in a document are independent (not only different words), aggravating the problem of exaggerated probabilities and systematically underestimating the distribution of term frequencies (number of occurrences of a term in a document) in real text — part of the motivation for the Poisson models and corrections to term frequency (Rennie et al. 2003). All document models inflate the probability of long documents if they are used with feature selection metrics that mainly select features predicting relevance, because long documents possess more features (Lewis 1998).

Lastly, biases in estimating the naïve Bayes parameters can affect performance. In the multivariate Bernoulli model of Section 2.2.3, the parameters θ_{ci} are the probability of feature i occurring in a document of class c. Using maximum likelihood to estimate θ_{ci} gives an estimate of N_{ci}/N_c: the number of documents in class c having feature i, divided by the number of documents in class c. However, the maximum likelihood estimate of θ_{ci} is biased and disrupts classification decisions. If by chance no training documents of class c have feature i, the maximum likelihood estimate gives test documents zero probability of being of class c when they possess feature i. Such features should count against the test document being of class c, but not rule out class c completely. Rennie et al. (2003) also observe a bias when the training data is skewed: a class having relatively few training examples (low prevalence) experiences greater under-estimation of its parameters than classes with relatively many training examples.

Smoothing methods make less biased parameter estimates, or at least avoid giving extreme 0/1 probabilities to test documents. With multivariate Bernoulli models, the generic approach to smoothing is to introduce a beta distribution prior on the θ_{ci} parameter. To the prior distribution, evidence from the training documents is added to yield a posterior distribution (also beta) for θ_{ci}. From the posterior distribution the most probable value of θ_{ci}, or maximum a posteriori (MAP) estimate, is obtained as (Fuhr 1992):

$$\hat{\theta}_{ci} = \frac{N_{ci} + a}{N_c + a + b} \tag{2.4}$$

where a and b are the parameters of the prior beta distribution. Laplace smoothing or

Laplace's rule of succession (McCallum and Nigam 1998, Lewis 1998) uses a flat prior distribution, obtained by setting $a = b = 1$, and results in a prior estimate for θ_{ci} of 0.5. As it turns, out a and b have a frequentist interpretation, because Equation 2.4 looks like a maximum likelihood estimate that includes a fictitious documents of class c in which term i occurred and b fictitious documents in which term i did not occur. For this reason a and b are also termed pseudocounts, and constitute "prior knowledge" (other than the training evidence from N_{ci} and N_c) about the Bernoulli distribution with parameter θ_{ci}. In Section 3.1.2 we show that when the training data is skewed (one class much more prevalent than the other), Laplace smoothing relatively over-estimates θ_{ci} for the low-prevalence class, radically biasing the binary classification feature weights in its favour. We then propose a correction for the aforementioned bias, designed for the multivariate Bernoulli model. For models like multinomial Bayes that use term frequency, the corresponding form of Laplace smoothing (Nigam et al. 2000) is often replaced with statistical language modelling methods such as absolute discounting, which redistribute probability from high frequency terms to lower-frequency ones (He and Ding 2007, Vilar et al. 2004).

2.3 Evaluation of classifier effectiveness

Given an initial corpus of documents whose class labels are already known, cross validation is commonly used to evaluate a classifier, training it and testing its predictions in a way which prevents overfitting phenomena from biasing the results. The result of cross validation is a list of either predictions or scores for each document in the initial corpus (divided up into test folds), from which many performance statistics can be derived. This section explains cross validation and overfitting, and evaluates different performance statistics, to identify caveats and the most useful metrics for evaluating binary classifiers that perform document-ranking on highly skewed data.

2.3.1 Cross validation

Cross validation is an extension of the train-and-test method of evaluating a classifier. Train-and-test shuffles the initial corpus, and then splits it into two parts, a large training corpus, and smaller test corpus. Training the classifier consists of providing the labelled

documents of the training corpus to the learning algorithm, which infers a classification rule $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ to decide whether the document is or is not a member of a given category. Testing the classifier consists of using Φ to make label predictions for the documents of the test corpus. Classifier predictions are evaluated by comparing them to those of the target function $\breve{\Phi}$ (Sebastiani 2002), which in practice is the human(s) who labelled the documents in the initial corpus. Because the judgements are subjective and not always consistent, the best a classifier can practically achieve is an error rate on the level of the disagreement between humans. Train-and-test however under-estimates performance if the training set is small and produces highly variable performance if the test set is small, although bootstrapping can be used to reduce the variability of small test sets, as in Wilbur (2000).

For corpora with hundreds (or more) documents, k-fold cross validation is the most common method of testing classifier predictions. First, the initial corpus is shuffled and divided into k segments — typically 5 or 10. For a binary classifier, the "relevant" and "irrelevant" parts of the initial corpus may be shuffled and divided separately to keep the same class ratio in each fold. Training is then performed using $k - 1$ segments, and testing is performed on the remaining segment or fold. The process is repeated k times to yields classifier predictions (or scores) on every document in the initial corpus. The test folds may first be aggregated and a single performance statistic calculated (micro-averaging), or performance calculated inside each test fold and averaged later (macro-averaging). The advantage of cross validation over train-and-test is that it can use most (90%) of the initial corpus for training, and in the end provides test predictions on all documents in the initial corpus. Using 90% of the initial corpus in validation is slightly pessimistic compared to using the whole initial corpus when training for operation. The logical extreme of k-fold validation is then leave-one-out, which learns a classifier from every document except the one whose class or score is being predicted. Leave-one-out uses the maximum amount of training data but is only suited to small corpora because it is much slower than cross-validation — even though some classifiers like naïve Bayes can be modified to "subtract" each test document in turn from a trained-on-everything classifier.

2.3.2 Overfitting

Elaborate designs like train-and-test, cross validation and leave-one-out are needed because classifiers overfit to their training data. Overfitting is when learning algorithms associate class labels with contingent (chance) characteristics of the training instances, not only the constitutive characteristics of the class itself (Sebastiani 2002). These chance associations lead a classifier to make better predictions on instances whose labels it saw during training than for unseen instances. All classifiers overfit, but the k Nearest Neighbours classifier with $k = 1$ provides an extreme case: it memorises the training instance locations and uses only the closest instance to make its decision, thus re-classifying training instances perfectly. In some classifiers, such as decision trees, overfitting can seriously degrade test performance and the trees must be "pruned" to reduce it. An unrelated phenomenon is over-training, which is when enough training data is available that adding more fails to improve performance. Overfitting is due to the "curse of dimensionality". When a vocabulary has many terms (feature vector $\mathbf{f}$ has many dimensions), training data only sparsely populates the space of possible instances, making some features appear predictive purely by chance. When features are many and examples are few, a classifier may end up considering "yellowness" to be predictive of something being a car. Validation designs prevent the classifier from overfitting to test documents, which would then inflate performance, by hiding the association between test document and their labels from the training step, which usually means hiding the test documents themselves.

Statistical feature selection (Section 2.4.3) tells the classifier which features to use, and so must also use only the training data available in each cross validation fold. Selecting features on data that includes labelled test documents is simply a different means of learning from the test data and should therefore be avoided. The specific form of overfitting in this case would be keeping particular features that are associated-by-chance to the labels of the test documents and would have been discarded had only the training data been used.

A subtle form of overfitting occurs with learning algorithms that perform tuning. Tuning is when the classifier modifies its internal parameters (such as misclassification costs, feature weights or decision thresholds) in response to performance. Contingent feature-class associations from tuning documents influence the classifier indirectly via the performance feedback, while associations in training documents influence the classifier directly. Test documents therefore should not be used for tuning. Rather, the training set should be

		Manual Judgement	
		Positive	Negative
Prediction	Positive	TP	FP
	Negative	FN	TN

Table 2.1: Confusion matrix tabulating the matches and mismatches between classifier judgements and manual judgements on the test data. The classifier judges each test item to be "positive" (member) or "negative" (non-member) of the class, and the classifier's prediction is "true" if it is the same as the manual judgement, and "false" otherwise. Test documents are thus divided into true positives (TP) and false negatives (FN) for members of the class, and true negatives (TN) and false positives (FP) for non-members of the class. The total $TP + FN + TN + FP$ sums to the number of test documents, of which $TP + FN$ had been manually judged to be members, and $TN + FP$ had been manually judged to be non-members.

further sub-divided to construct a "hold-out" set of tuning documents (Sebastiani 2002). The learning algorithm then trains on (and overfits to) the reduced training set, the tuning algorithm tunes on (and overfits to) the hold-out set, and the tuned classifier finally makes test predictions on the test set. Hold-out has been used to tune SVMs for detecting abstracts on drug-drug interactions (Duda et al. 2005) and evidence-based medicine (Aphinyanaphongs et al. 2005).

2.3.3 Precision and recall statistics

For a given class, performance statistics of a hard classifier are derived from the confusion matrix of Table 2.1, which is a contingency table of classifier predictions against manual judgements. "Positive" and "negative" refer to membership and non-membership of the class, and because binary classification focuses primarily on the "relevant" class, with the irrelevant class being complementary, "positive" and "negative" are often used interchangeably with "relevant" and "irrelevant" respectively. With more than two classes, the performance over all classes may be micro-averaged by summing the confusion matrices of the different categories and calculating a single metric; or the performance may be macro-averaged by calculating the metric for each category's confusion matrix and then taking the mean. With micro-averaging, rare categories contribute less than common ones, while with macro-averaging rare and common categories contribute equally to the final statistic.

In machine learning, classifiers are often judged on accuracy, the fraction of all predictions

that were correct: $(TP + TN)/(TP + FP + TN + FN)$. However, a trivial rejector that predicts "everything is irrelevant" has high accuracy if documents irrelevant to the topic in fact make up the vast majority of the corpus. Metrics from information retrieval are more informative with skewed test data. For instance, precision or positive predictive value is the fraction of predicted relevant documents that were in fact relevant: $\pi = PPV = TP/(TP + FP)$. Recall or true positive rate is the fraction of all relevant documents retrieved, $\rho = TPR = TP/(TP + FN)$. Precision π and recall ρ may be combined by taking the harmonic mean, called the F_1 measure, $F_1 = (0.5\pi^{-1} + 0.5\rho^{-1})^{-1} = 2\pi\rho/(\pi + \rho)$.

The confusion matrix is designed for classifiers that make hard classification decisions. For a classifier that predicts documents scores and rank them by decreasing score, the confusion matrix becomes a function of the chosen score threshold for classifying a document as relevant. The relationship is illustrated in Figure 2.2: lowering the threshold retrieves more documents, raising both TPR and FPR. Using precision and recall statistics again, one can plot the relationship of precision to recall in a precision-recall curve like that of Figure 4.5, where lower decision thresholds correspond to lower precision and higher recall. The inherent trade-off means that biomedical database curation, which calls for high recall, must necessarily accept somewhat lower precision. Summarising the precision-recall curve is achieved using averaged precision. If the initial corpus is ranked by decreasing document score, averaged precision is calculated by averaging the precision values over each rank where a relevant document occurs (sum of precisions divided by the number of relevant documents). Another way to summarise the curve while retaining more information about its structure is to use the 11-point interpolated precision, which consists of the interpolated precision at recall equal to $0, 0.1, 0.2, \ldots, 1$. Interpolated precision is defined to be the largest value of precision at that recall or any higher recall (Manning et al. 2008; Chapter 8). This is because precision generally decreases with greater recall, but jumps when relevant documents are encountered, making the curve somewhat jagged and meaning higher precision can sometimes be achieved by taking a few more results.

2.3.4 The Receiver Operating Characteristic

The Receiver Operating Characteristic (ROC) (Fawcett 2006) compares the two "operating characteristics" of sensitivity $TPR = TP/(TP + FN)$ and false positive rate $FPR = FP/(TN + FP)$ by plotting TPR against FPR as the decision threshold varies in

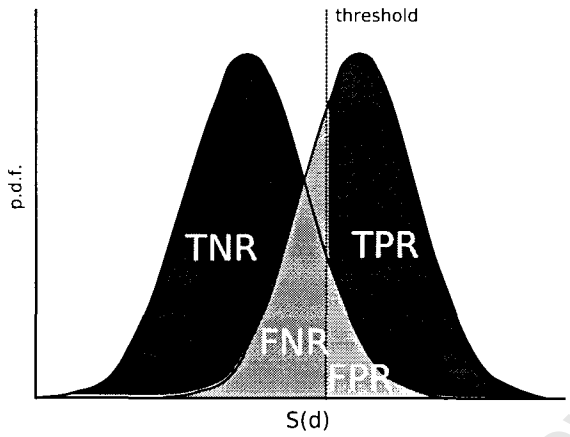


Figure 2.2: Distributions of classifier scores for relevant and irrelevant articles, illustrating how setting a score threshold on a document-ranking classifier is used to obtain the confusion matrix of hard classification. Each curve represents a probability distributions (probability density function), the left curve being negative documents (non-members) and the right curve being positive documents (members of the class). The classifier predicts documents below the threshold line to be negative, which therefore constitute the true and false negatives. Documents at or above the threshold constitute the true and false positives. The areas under curve to the left and right of the threshold represent the rates of true and false negatives and positives.

Figure 2.2. The value of ROC in diagnostic testing is that it evaluates the trade-off between sensitivity and specificity, where sensitivity is equivalent to recall, and specificity is the true negative rate (where $FPR = 1 - TNR$). The area under curve (AUC) of the ROC is a useful summary statistic. Hanley and McNeil (1982) showed that the AUC as calculated by trapezoidal rule (a slight under-estimate) is also the probability of correctly ranking a randomly chosen relevant document above a randomly chosen irrelevant document — in turn equalling the Mann-Whitney sum-of-ranks statistic U — and provided a tabular method for calculating its standard error (Hanley and McNeil 1982; Table 2). Several studies have used the ROC AUC to compare classifiers of biomedical text (Aphinyanaphongs et al. 2005, Wang et al. 2007, Duda et al. 2005).

The ROC has been recommended over accuracy in machine learning for its statistical properties (Bradley 1997) and because, all else being equal, it is independent of class skew (Fawcett 2006). Unlike the accuracy statistic, trivial classifiers that reject/accept everything or rank documents randomly have poor ROC curves no matter how skewed the classes. Medline is a prime example of class skew, with a very low prevalence of

relevant documents for a given topic. ROC area is therefore robust for ranking different classifiers on the same data, or for ranking the difficulty of different data sets under the same classifier. ROC has nonetheless been criticised for being "overly optimistic" when evaluating a classifier on highly skewed data (Davis and Goadrich 2006). To illustrate, compare Medline to more balanced data sets like those of Aphinyanaphongs et al. (2005) or Wang et al. (2007), where Medline has been pre-filtered to create a small collection whose members were manually labelled to create the initial corpus. Pre-filtering has two effects: less class skew (due to higher prevalence of relevant documents); and greater similarity between irrelevant and relevant documents than in Medline at large (due to sharing of keywords). The second effect makes it much easier to rank a random relevant/irrelevant pair drawn from Medline (which is highly skewed) than a pair drawn from a pre-filtered subset (which is balanced). ROC area thus measures first the inherent difficulty of ranking the pair, and secondly the effectiveness of the classifier, with high-skew data presenting the easier ranking problem. Another problem with ROC is that in a large data set like Medline there are so many documents that only the lowest values of the FPR are relevant to the results. Even 0.1% false positives means 16,000 irrelevant results, yet the ROC area summarises the curve all the way up to 100% false positives. Lastly, ROC obscures the degree to which one classifier is better than another, and over what ranges of decision threshold — for which purpose Drummond and Holte (2006) introduces "cost curves", a more complex performance analysis, that in the simplest case where a false positive costs the same as a false negative, transforms ROC into a plot of best possible error rate against the prior probability of relevance. Overall, ROC area is a good comparative metric, but a poor indicator of absolute performance under high class skew.

Precision-recall curves have been recommended over ROC for judging absolute performance under high class skew and for comparing different classifiers on the same data set (Davis and Goadrich 2006). Precision (along with recall) directly indicates retrieval performance, and tuning the classifier to optimise ROC area does not necessarily optimise the averaged precision in the precision-recall curve (Davis and Goadrich 2006). However, unlike ROC, precision is a direct function of the class skew, making it less suited to comparing the difficulty of differently-skewed corpora for the same classifier, including extrapolating from performance on a test set to real-world data that has a different frequency of relevant documents. Attempting to extrapolate precision from test performance to differently-skewed operational data is a common mistake in classifier evaluation (Jensen et al. 2006), also observed in methods listed in Section 2.6. The dependence of precision on class skew

arises from precision being a posterior probability of relevance (given that the classifier predicted the document relevant), which therefore depends on the prior probability of relevance. This prior probability is the frequency or prevalence of relevant documents in data, which is also the precision of the trivial acceptor (which predicts every document to be relevant). Historically, biomedical text classification has been evaluated on corpora with a high proportion of relevant documents, in contrast to Medline at large which has 0.1% or lower prevalence of relevant documents depending on the breadth of the subject domain. The relationship of precision to prevalence for a given value of TPR and FPR is as follows, where r is the prevalence in a corpus of size N:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TPR \cdot r \cdot N}{TPR \cdot r \cdot N + FPR \cdot (1 - r) \cdot N} = \frac{TPR}{TPR + \frac{1-r}{r} FPR} \quad (2.5)$$

2.3.5 Stochastic variation in performance

Focusing on document-ranking binary classifiers, several groups of factors interact in generating the value of a performance statistic, including: the constitutive characteristics of the relevant and irrelevant class, the chance characteristics from sampling the train and test sets, and the constitutive characteristics of the classifier. Stochastic variation or uncertainty in performance arises from sampling. Take, for example, an ROC curve derived from document scores in some initial corpus, which in turn were obtained by aggregating the document scores from the test folds of a cross validation experiment (micro-averaging the folds). Because the area under the ROC curve is a Mann-Whitney U statistic, the standard error of its sampling distribution can be calculated from the data, which allows the use of the Z-test to detect whether two initial corpora are significantly different, under the null hypothesis of both initial corpora being independently sampled from the same distribution or universe of documents (Medline or a Medline subset in this case). The Z-test however has low power to detect whether two classifiers have significantly different performance on the same sample, because it assumes sample variation where there is none. For greater power to detect differences between classifiers, one would use some form of paired permutation test for the null hypothesis of equal area under curve (or of identical curves), which permutes the relevant/irrelevant labels of the initial corpus by shuffling (Braun and Alonzo 2008) or permutes which classifier was used by swapping the document scores produced by

each. Significance tests for ROC difference are used in Aphinyanaphongs et al. (2005) and Wang et al. (2007).

Other statistics, like the averaged precision of a ranking classifier or accuracy of a hard classifier, do not correspond to a statistic with known sampling distribution, so micro-averaging across validation folds (or using leave-one-out validation) gives only a point estimate. However, using macro-averaging to calculate the statistic within each test fold, produces 10 point estimates from independent test samples (the test folds), from which one can report the mean and range across the validation folds, as in Aphinyanaphongs et al. (2005) for ROC area calculated within the test folds of 5-fold validation. Because by central limit theorem the sampling distribution of the (macro-averaged) mean is approximately normal with enough samples, one can also estimate the standard error of the estimate. However, it would have low power to detect differences between classifiers on the same sample, which is perhaps why significance testing is unusual in information retrieval. The TREC competition (Hersh et al. 2005) for instance uses a train-test design, and simply ranks retrieval-classifiers by averaged precision, where one can see whether a difference is substantial (as opposed to significant). In large multi-label problems where one classifier may outperform the other at predicting some of the classes but not others — differences which would be obscured by micro- or macro-averaging the results — there are however tests for whether one classifier is better overall (Yang and Liu 1999).

When comparing different classifiers using cross validation on the same initial corpus, ensuring that the train/test splits are the same in each cross validation fold requires setting the random number generator seed before shuffling the initial corpus. The same seed produces the same shuffle each time, so cross validating the same classifier twice will produce exactly the same result. If different shuffles were used, performance would vary because the train/test splits were different, but it would not be independent sampling, because the test samples would be drawn from the initial corpus instead of from the domain from which the initial corpus was sampled. The resulting shuffle variance in performance would therefore be smaller than that of independent samples. Keeping cross validation test splits identical between runs (no shuffle variance) also avoids the chance of one classifier getting an unusually "easy" shuffle with above-mean performance while the other gets a more difficult one.

2.4 Representation of Medline records

Many classifiers operate on vector representations of documents where each element indicates the importance of a vocabulary term to the document. The conversion of documents into vector representation is text indexing, a sub-problem shared by supervised learning and information retrieval that uses methods from natural language processing. Text indexers split text into tokens (usually words), and the "bag of words" for each document is then easily converted into a feature vector. Indexing methods often pre-process text to remove uninformative terms and cluster synonyms. After indexing, other algorithms select and cluster features based on the feature vectors of the training data. We focus on the caveats raised in Medline records, considering the characteristics of biomedical text, ways of indexing the structured information in Medline records, and the use of Medical Subject Headings — a resource unique to Medline that condenses the major topics of the full-text article.

2.4.1 Text indexing for biomedical abstracts

Tokenisation breaks text into lexical units (tokens or terms), and words are the usual lexical unit for text classifiers. Some applications may also generate tokens for punctuation (Gospodnetic and Hatcher 2005), and label words with parts of speech, named entities, or syntactic phrases using NLP libraries such as LingPipe (LingPipe 2008). Regarding phrases, they may be identified either using syntactic rules, or statistically from frequently occurring pairs or sequences of words. Phrases carry more meaning than isolated words (compare "calcium" and "calcium channel blocker"), but Lewis (1992) found them less suitable for classification, requiring more terms for a given level of effectiveness and having a lower maximum effectiveness. The statistical problem with phrases is partly due to noise from having lower frequencies in text, and partly due to ambiguity. Synonyms have nearly the same meaning, while homonyms are spelled the same but express different meanings depending on the context. Synonymy and homonymy reduce the information that each term is able to provide about the class variable, and are far more prevalent amongst phrases than individual words. Interestingly, one use for classification is word sense disambiguation, which uses the text surrounding a word to distinguish, for example, between different meanings of the word "bank" (Sebastiani 1999). Metzler and Croft (2005)

indicates that dependency information and phrases are more useful on larger training sets and longer documents. Yetisgen-Yildiz and Pratt (2005) however found that prediction of MeSH terms from short article titles in the OHSUMED Medline corpus (Hersh et al. 1994) improved with a hybrid phrase/word approach, and Wilbur (2000) found that including pairs of consecutive words improved effectiveness at filtering biomedical texts so long as feature selection was used to retain only informative terms.

When tokenising for word features, text indexers often transform the text to prune uninformative words and cluster synonymous words. Stopword removal eliminates common words such as "the" and "very" that in isolation contain little information about the class of the article. Stopword removal originated in information retrieval, but is no longer widespread there because users regularly search for phrases that include stopwords. Other transformations include case-folding, which converts the text to lower case so that differences in case are ignored, and stemming, which strips word endings to convert inflections of a word (expressing plurality or tense) to a common morphological root. These methods select and cluster features directly from the text, differing from the statistical approaches in Section 2.4.3 that use the feature vectors. The usual stemming algorithm is that of Porter (1980), or some variant (Porter 2001). Stemming is essential in information retrieval because users enter one form of a word in a query but intend all forms to be detected, but its use is controversial in text classification (Sebastiani 2002). In at least one case stemming reduced effectiveness in classifying Medline records (Wang et al. 2007), possibly because the rules for plain English inappropriately transform biomedical terminology. Also, Riloff (1995) finds that stopword removal and stemming are counter-productive when basing classification on phrase-like "relevancy signatures", because auxiliary words and the form of a word may then carry information about the class variable. The popularity of stemming has however led to it being used without appraisal in some biomedical text classifiers, such as one for the filtering task in the TREC Genomics Track (Cohen 2006) and one for detecting clinical practice articles (Aphinyanaphongs et al. 2005). Another clustering transformation considers all occurrences of numbers to be occurrences of an artificial "number" token, which can be valuable if numbers are more prevalent in documents from some categories than others. Although biomedical texts may not benefit from number clustering, the principle has been applied to clustering of named entities in particular domains. Wang et al. (2007), for example, improved the detection of articles about immune epitopes by using regular expressions to detect occurrences of peptide sequences such as "SHNFENKL" or MHC alleles such as "HLA A*0201" and replacing all such occurrences with the artificial

tokens "∼peptide∼" and "∼mhc_allele∼" respectively.

Having extracted the features of documents, the documents are usually represented as feature vectors as in Section 2.2.3, although in text classification it is usual to list only the non-zero members of the feature vector as most features (vocabulary terms) do not occur in a given document. The vector representation is also known as "bag of words" because it ignores the order in which terms occur in the document, representing only the lexical semantics and not the compositional semantics of the text (Sebastiani 2002). The problem of capturing compositional semantics is related to the problem of term dependency, where the occurrence of one term in a text (such as "mouse") makes certain other terms (such as "liver") more likely to occur, either in the sentence or the complete document. Friedman et al. (1997), for example, models co-occurrence dependency between pairs of terms, and Peng and Schuurmans (2003) and Metzler and Croft (2005) model the sequential dependence of terms using Markov models in which the probability of occurrence of a term depends on the neighbouring terms.

2.4.2 Structured information as a source of features

Structured documents and Medline records

Historically, text classification has been performed on unstructured documents such as news articles or Internet discussion group postings. Increasingly, structured documents are becoming available, such as the XML-formatted papers available from the BioMed Central open-access publishing house (BMC 2008) and the NLM's PubMed Central repository (PMC 2008). Structured texts explicitly represent sections and titles, and may also identify named entities such as genes and proteins. Medline records, such as the example in Figure 2.3, consist of an abstract and bibliographic metadata rather than full text, but provide structured information in the form of record fields, such as title, abstract, journal, authors, publication date, and MeSH terms (NLM 2007e). Text indexers for classification often ignore structure, reducing structured documents to raw text before indexing. This is the case in Wang et al. (2007), while Aphinyanaphongs et al. (2005) incorporates some structure by prepending with "mh_" to complete MeSH phrases, and prepending "title_" to words from the title. In Section 3.2 we describe the use of separate feature spaces and specialised extraction of features for each of the Medline record fields, reserving ordinary

```
PMID -  8372948
OWN -  NLM
STAT -  MEDLINE
DA -  19931012
DCOM -  19931012
LR -  20041117
PUBM -  Print
IS -  0147-5185 (Print)
VI -  17
IP -  10
DP -  1993 Oct
TI -  Primary angiosarcoma of the spleen. A clinicopathologic study of 40 cases.
PG -  959-70
AB -  Forty primary splenic angiosarcomas occurring in 21 men and 19 women...
AD -  Department of Hematologic and Lymphatic Pathology...
FAU -  Falk, S
AU -  Falk S
FAU -  Krishnan, J
AU -  Krishnan J
FAU -  Meis, J M
AU -  Meis JM
LA -  eng
PT -  Journal Article
PL -  UNITED STATES
TA -  Am J Surg Pathol
JT -  The American journal of surgical pathology
JID -  7707904
RN -  0 (Biological Markers)
SB -  IM
CIN -  Am J Surg Pathol. 1995 Jan;19(1):119-20. PMID: 7802132
MH -  Adult
MH -  Aged
MH -  Aged, 80 and over
MH -  Biological Markers/analysis
MH -  Female
MH -  Hemangiosarcoma/chemistry/*pathology/physiopathology
MH -  Humans
MH -  Immunohistochemistry
MH -  Male
MH -  Middle Aged
MH -  Splenic Neoplasms/chemistry/*pathology/physiopathology
EDAT -  1993/10/01
MHDA -  1993/10/01 00:01
PST -  ppublish
SO -  Am J Surg Pathol. 1993 Oct;17(10):959-70.
```

Figure 2.3: Medline record for PubMed ID 8372948 in the older MEDLINE format (NLM 2008a), with long lines truncated. See Figure 3.1 for the same record in the newer Medline XML format.

word-based text indexing for the title and abstract.

Medical Subject Headings in particular provide a unique source of information about the topics of an article. Medline records possess an average of around 10 terms, each provided as a MeSH descriptor in association with zero or more qualifier subheadings. Subheadings marked with a "*" signal major topics of the article. For example, examining the "MH" fields of Figure 2.3 indicates that pathology of splenic neoplasms and hemangiosarcoma are major topics of the article. The descriptors are drawn from the MeSH controlled vocabulary (NLM 2007e), which is arranged in a tree-like structure of concepts, with specific descriptor terms at the bottom — although each concept or term may appear in more than one branch of the tree. There are 24,767 descriptors and 83 qualifiers in the 2008 MeSH. MeSH terms are assigned manually by NLM curators based on the full text of the article. The Medline Indexing Initiative (Aronson et al. 2000) has however recently developed the Medical Text Indexer (Aronson et al. 2004, Gay et al. 2005), which uses machine learning on titles and abstracts to assist human indexers in *semi-automatic indexing*, speeding up the process and improving recall of terms, since it is difficult for human indexers to identify all relevant terms, even with the aid of the MeSH thesauri of 172,000 supplementary concept records and 97,000 term synonyms.

The value of MeSH in classification

Kostoff et al. (2004) evaluated the information content of title and MeSH terms compared to abstract in the context of a particular medical condition (Raynaud's phenomenon). The title (without the abstract) was found to produce a sparsely populated feature space, having a large vocabulary relative to the number of occurrences of terms, which is the same statistical problem experienced with phrases. This fact supports the concatenation of title and abstract, which is common in classification studies (Wang et al. 2007, Lee et al. 2006, Aphinyanaphongs et al. 2005, Bartling et al. 2003). Wang et al. (2007) also found that appending author and journal names to the text of the abstract improved classifier effectiveness. In Section 4.1.3, we directly evaluate the value of the title+abstract, MeSH, journal or author list feature spaces on different topics.

As a controlled vocabulary with a preferred term for each concept, MeSH terms specify topics present in the full text of the article without the ambiguities of text words, where

synonymy and homonymy partially confounds the mapping between terms and concepts. The MeSH vocabulary is also much smaller than that of text words, so the space of possible documents is more densely populated, which makes learning a classification rule easier and reduces the need for feature selection. The literature is however mixed regarding the value of MeSH terms in classification. Kostoff et al. (2004) observes that, being controlled, MeSH lags current terminology, Medline records take 1–3 months for terms to be assigned, and MeSH terms tend to be less specific about the topic of the article than phrases in the abstract. In coverage of topics, the less "biomedical" a concept is, the less finely it is resolved in the MeSH vocabulary (NLM 2008c). In an extreme case, MeSH has only one term "automobiles" for all wheeled forms of transportation, but has distinct terms for different preparations of aspirin. The value of MeSH in classification is therefore strongly dependent on the subject domain. Finally, Bartling et al. (2003) noted that the term "dental research" tended to be applied to articles discussing the topic of dental research, rather than dental research articles themselves. The MeSH vocabulary sometimes makes this distinction explicit, for example in the terms "Clinical Trials, Phase II" (actual clinical trials) and "Clinical Trials, Phase II as Topic" (papers which just discuss clinical trials).

Regarding the effectiveness of MeSH in a classifier of Medline records, Wang et al. (2007) found that appending MeSH terms to the abstract text improved effectiveness on their data set. Doing so however uses only individual words, which may only partially capture the concepts of multi-word MeSH terms. Rubin et al. (2005) found that MeSH outperformed title/abstract as a source of features for their data set when only the top 150 features were selected. Aphinyanaphongs et al. (2005) however found that including MeSH made no difference over title/abstract on their data, and Bartling et al. (2003) found that adding MeSH terms reduced performance with their method and data. The precise method of using MeSH features may be important: Wang et al. (2007) and Rubin et al. (2005) split up MeSH descriptors from their qualifiers to reduce the feature space, while Aphinyanaphongs et al. (2005) kept them together and recognised that doing so may have been why using MeSH did not improve performance. Fewer documents would have the compound feature "Splenic Neoplasms/chemistry/pathology/physiopathology" than would have each term separately, reducing the number of articles that the feature can help classify. Bartling et al. (2003) does not specify how MeSH features were used with the Identify Patient Sets (IPS) (Aronis et al. 1999) classifier, and possible influences include IPS mapping its texts onto Unified Medical Language System (UMLS) terms (NLM 2006c), and being designed for retrieving patient records using interactive construction of a classifier and query expansion.

2.4.3 Dimensionality reduction and feature selection

When documents are represented as feature vectors $\mathbf{f} = (f_1, \ldots, f_{|\mathcal{V}|})$, the value $|\mathcal{V}|$ is dimensionality of the vector space in which the documents occur, which is the size of the vocabulary of terms found in the documents. The MeSH feature space, for example, would have just under 25,000 terms, while the space of title/abstract words could have many millions of terms. Particularly in biomedical text, many of these terms may be non-dictionary sequences of letters and numbers such as genes and chemicals. The speed and effectiveness of the naïve Bayes classifier discussed in Section 2.2.3, is relatively independent of feature space size, but neural networks and other classifiers in Section 2.2.3 are intractable, slow or perform poorly with large feature spaces. A smaller vocabulary of features with more predictive power may reduce overfitting (Section 2.3.2), where a classifier "is tuned to the contingent characteristics of the training data rather than just the constitutive characteristics of the categories" (Sebastiani 2002). Dimensionality reduction algorithms transform $\mathcal{V}$ into another feature space $\mathcal{V}'$, where the dimensionality $|\mathcal{V}'|$ is much smaller than $|\mathcal{V}|$. The aggressiveness of the reduction is $|\mathcal{V}|/|\mathcal{V}'|$ (Sebastiani 2002).

Dimensionality reduction may be achieved either through term extraction, or feature selection. Feature selection chooses a subset of the original features to weed out uninformative features, while term extraction maps the original features onto a new feature space such that near-synonymous features are combined. Homonyms are generally not disambiguated. Unsupervised learning can be used to perform term extraction by grouping terms into clusters to form the reduced feature space. Stemming during tokenisation effectively clusters or groups features using linguistic rules, but "term clustering" usually refers to methods of grouping features based on their *meta-features*, such as the documents in which each term appears (Lewis 1992). A non-clustering term extraction method is Latent Semantic Indexing (Deerwester et al. 1990) which uses matrix algebra to create a new feature space from weighted combinations of the original features. Term extraction can increase frequency of the new features, reducing problems with data sparseness (too few training examples).

Feature selection involves assigning some score to each term, and choosing either a fixed number of features from the top-ranked terms or a score threshold. Up to a 100-fold reduction in feature space may be possible with little loss in performance, and less aggressive reductions may even improve performance in certain cases (Yang and Pedersen 1997). A range of feature selection metrics are presented and compared in Yang and Pedersen (1997)

and Forman (2003). The simplest and most widely used is document frequency, the number of documents in which a term appears. Wang et al. (2007) for example selects only features that occur in at least 4 documents. Zipf's law observes that the probability of a term having a particular document frequency drops exponentially as document frequency rises, with the vast majority of terms occurring in only one or a few documents in the collection. The lack of training instances for those lowest-frequency terms generally makes them poor class predictors, and each such excluded term affects a few documents at most. On the other hand, low-to-medium frequency terms are some of the most informative (Sebastiani 2002) — a corollary of the fact that a term occurring in most documents is usually a poor predictor of class (unless as part of a phrase as in Riloff (1995)). Despite its simplicity, Yang and Pedersen (1997) found that document frequency holds up well to more computationally expensive metrics such as information gain.

Information gain or average mutual information is a widely used feature selection metric, which we also make use of in Section 3.2.3. In machine learning information gain was first used with decision tree classifiers, and in that context is defined as the "expected reduction in entropy caused by partitioning the examples according to the attribute" (Mitchell 1997; page 58). Suppose there are $|\mathcal{C}|$ classes, and C is the random variable over $c \in \mathcal{C}$. Following the notation of Equation 2.3, let F_i be a random variable for absence or presence of the ith vocabulary term in a document, taking on values $f_i \in \{0, 1\}$. The Shannon entropy of a discrete random variable C is $H(C) = -\sum_{c \in \mathcal{C}} P(c) \log_2 P(c)$. The difference in entropy between C and C conditioned on F_i then becomes (McCallum and Nigam 1998):

$$\begin{aligned} IG(C, F_i) &= H(C) - H(C|F_i) \\ &= -\sum_{c \in \mathcal{C}} P(c) \log_2 P(c) \\ &\quad + \sum_{f_i \in \{0,1\}} P(f_i) \sum_{c \in \mathcal{C}} P(c|f_i) \log_2 P(c|f_i) \\ &= \sum_{f_i \in \{0,1\}} \sum_{c \in \mathcal{C}} P(c, f_i) \log_2 \left(\frac{P(c, f_i)}{P(c)P(f_i)} \right) \end{aligned} \tag{2.6}$$

Issues in feature selection include the choice of metric and how to choose features using a given metric. In multi-class problems dimensionality reduction may either be global, with

one feature space for all categories, or local with a different feature space for each category (Sebastiani 2002). Using information gain (or other metrics) for global feature selection, Forman (2004) observed that the existence of "easy" classes having an abundance of predictive features can reduce performance on "difficult" classes whose predictive features are weaker and remain unselected. The solution in Forman (2004) is to set up m "one against all" binary classification problems and construct a global feature space by selecting the features with the greatest information gain in each binary problem in a round robin or randomised fashion — so that the classes are equally represented.

Binary classification of Medline records exhibits high class skew, with a very low proportion of records being relevant for a particular topic. Mladenic and Grobelnik (1999) compared feature selection metrics using a naive Bayes classifier under high class skew and found that an odds-ratio metric (as in Section 3.1.1) performed best on a Web-page corpus. Information gain is a one-sided metric, always being of positive sign, while log odds ratio is two-sided, with positive-scoring features indicating relevance and negative-scoring features indicating irrelevance. Because log-odds is two-sided and performed well, Mladenic and Grobelnik (1999) recommended simply selecting features that are found in relevant documents. However, two-sided metrics and ignoring "negative" features favours long documents because they have more features and thus appear more relevant just for being long, and also makes it more difficult to eliminate false positives by rejecting irrelevant documents with confidence, reducing performance (Zheng et al. 2004). Zheng et al. (2004) advocates choosing the number of features that indicate relevance and the number that indicate irrelevance for optimal performance, a form of tuning (Section 2.3.2). Most studies such as Yang and Pedersen (1997), Forman (2003; 2004) consider one-sided metrics with appropriate threshold adequate for binary problems whether balanced or imbalanced.

2.5 Related work on Medline filtering

Within the general problem of identifying Medline records relevant to some topic, different topics and retrieval needs motivate different solutions to the problem. For examples, individual researchers can enter query strings into relevance-ranking or boolean retrieval methods from Section 2.1 to locate specific documents, and related articles help when exploring the literature. However, databases whose curators extract information from the

literature have a different problem: that there is not one narrow topic to search for, but a family of topics with potentially thousands of keywords that might indicate that a Medline record is relevant. Historically, knowledge engineering in the form of large boolean queries has been used to filter literature, but methods based on supervised learning and text mining have been becoming more prevalent. Cohen and Hersh (2005) also identify a strong need for useful ways of applying text classification to biomedical literature.

We have developed a supervised learning solution to meet the objectives set out in Section 1.1: being effective across a wide range of topics, available as a retrieval service, filtering all of Medline, and being exceptionally fast. Below, we examine related research that also use supervised learning to identify Medline records in circumstances where many features potentially indicate relevance. The methods tend to have somewhat different objectives, being designed for the topic of a particular database. They are therefore not packaged as a service, may operate on a small pre-filtered subset of Medline or perform once-off classification of a larger subset, and so also have less need for computational efficiency.

2.5.1 Classifiers for biomedical databases

The Immune Epitope Database

The Immune Epitope Database (IEDB) records the structures of molecules that bind to major histocompatibility complex molecules or are recognised by the T and B cells of the human immune system (Peters et al. 2005a;b), and a range of intrinsic and extrinsic features of the epitope and its host environment. Most of this information is manually curated from scientific papers. Curators would manually review the results of many sensitive PubMed queries on sub-topics relevant to the IEDB, such as the one in Figure 2.1. Wang et al. (2007) set out to improve the precision of the knowledge-engineering results using supervised learning. First, the combined result sets of the sensitive queries (with date ranges to freeze the results) yielded a Medline subset of 20,910 records that had already been manually reviewed, forming a "gold standard" training corpus of 5,712 relevant and 15,198 irrelevant documents. The text indexer at first extracted only word features from title and abstract of the Medline records, with no feature selection or other pre-processing. Using Weka (Frank et al. 2004), a naïve Bayes classifier outperformed an SVM, a decision tree and a perceptron on the ROC AUC statistic with 10-fold cross validation and micro-

averaging over folds. Feature extraction, lack of feature selection, and lack of parameter tuning may, however, have handicapped some of the classifiers more than others. With the naïve Bayes classifier, performance improved when MeSH terms, journal name and author list were concatenated onto the title/abstract text. Performance further improved by selecting only features occurring in at least 4 documents and having an information gain of at least 2×10^{-5} bits. The indexer removed stopwords, but there was no case folding, and Porter stemming was found to degrade performance. A domain-specific feature clustering step detected peptide sequence and MHC alleles and other features in the text, and created artificial clustered features such as "mhc_allele". The final classifier is expected to increase the concentration of relevant articles in the records examined by manual reviewers, that are then passed on to the curators. Notably, the evaluation corpora for the IEDB classifier were published as supplementary data to encourage their use in benchmarking. We therefore use the IEDB corpus in addition to our own in Chapter 4, and make a direct performance comparison in Section 4.3.

The Pharmacogenetics Knowledgebase

Another database, the Pharmacogenetics Knowledgebase (PharmGKB), collects information about interactions between pharmaceutical drugs and variations in human genes that influence adverse effects, rate of clearance, toxicity levels, and other clinically relevant properties (Hewett et al. 2002). Relevant documents describe "gene-drug interactions", and PharmGKB keeps track of what gene-drug interactions each article provides evidence for, providing a ready source of PubMed IDs since the vast majority of evidence articles are found in Medline. Few evidence articles are however indexed with the MeSH term "pharmacogenetics", which seems to be applied mainly to articles that discuss pharmacogenetics explicitly. There is also a method for detecting co-occurrences of gene and drug names in sentences (Chang and Altman 2004, Chang et al. 2004). Curation of the database has been oriented around community submissions (Rubin et al. 2004), so all that is available are relevant examples. There is no PubMed query like that of the IEDB to provide a pre-filtered subset of Medline on which the classifier may operate, making classifying all of Medline necessary.

A PharmGKB study evaluated several classifiers for the task of detecting gene-drug interaction evidence articles (Rubin et al. 2005). Features comprised abstract/title words,

MeSH terms, or both, and selection was according to the chi-square statistic on the contingency table of documents being relevant or irrelevant document versus features being absent or present. With 150 or 350 features (abstract words, MeSH terms, or both), logistic regression had substantially higher F-measure than naïve Bayes in Weka for natural decision thresholds. Using all features was not feasible in Weka, but a custom-built "log-likelihood" classifier that used all features slightly outperformed a logistic regression classifier that used the 350 best word/MeSH features. The classifier is like a naïve Bayes binary classifier in log-odds form, but discards feature absence (Equation 2.3 would only evaluating terms where $F_i = 1$), and instead of a prior score uses a manually chosen log odds threshold of 10, and uses maximum likelihood parameters with zero probabilities replaced by 10^{-8} for smoothing. Applying the classifier to Medline (possibly trained on 426 relevant and 9,722 irrelevant documents) yielded 11,922 results above the threshold. These were filtered for sentence co-occurrence of genes and drugs to yield 4,892 documents, of which a pharmacologist judged that 92% of the top 1,649 documents were relevant.

The TREC Genomics filtering task

The 2004 and 2005 Text Retrieval and Evaluation Conference (TREC) Genomics Track (Hersh et al. 2005) featured a "document triage" task using supervised learning to identify documents relevant to the Mouse Genome Database (Eppig et al. 2005). There were four sub-topics: tumour biology, embryologic gene expression, Gene Ontology annotation (Consortium 2008) and alleles of mutant phenotypes. Classifiers were trained on full-text articles from three journals from 2002 and tested on 2003 articles from the same journals, with the training and test data pre-filtered to hold only articles mentioning some form of the word "mouse". This task ranked classifiers on an overall "utility" score, which gives a classifier U_r points for each true positive, and subtracts one point for each false positive. Cohen (2006) provides a method of optimising a support vector machine for the TREC Genomics task. The TREC task differs from the problem approached here by dealing with full-texts instead of abstracts and by operating on a concentrated test corpus containing a large proportion of relevant documents instead of a broad test corpus in which relevant documents are rare.

2.5.2 Classifiers for protein interaction abstracts

The Biomolecular Interaction Network Database (BIND) (Alfarano et al. 2005) collates information about molecular interactions, originally protein-protein interactions. PreBIND (Donaldson et al. 2003) provides a resource to BIND curators, and uses supervised learning to detect articles, followed by information extraction using a dictionary of genes and proteins to detect co-occurrences of the same within particular sentences. PreBIND uses a non-linear SVM (radial basis function kernel) trained on a small manually constructed collection of 693 relevant documents (describing a biomolecular interaction) and 401 irrelevant documents with similar word uses. Pre-processing removed stopwords, folded case, removed all non-alphabetic characters, and truncated strings to 10 characters. The indexer considered single and two-word phrases. The SVM achieved a precision-recall break-even of 92%, outperforming a multinomial Bayes classifier which obtained 87%. However, the Bayes classifier only used the 100 features with highest information gain following the choice in Marcotte et al. (2001) — but other researchers showed that using so few features substantially reduces performance (Rubin et al. 2005). After evaluation, the same trained SVM was applied to a larger subset of Medline (1.88 million records) and predicted 269,000 relevant documents. It was expected to take 3 days if applied to all of Medline. Donaldson et al. (2003) however indicated that the 92% precision on the test data (in which relevant documents outnumber irrelevant ones) would also be observed when classifying Medline, which is incorrect: the drastically lower prevalence of relevant records in Medline will reduce precision, as discussed in Section 2.3.4.

Curators for the Database of Interacting Proteins (DIP) (Xenarios et al. 2002), which curates information related to protein-protein interactions (PPI) in yeast, also developed a classifier(Marcotte et al. 2001). The classifier was naïve Bayes with a Poisson distribution modelling the occurrences of each word in the relevant versus irrelevant class, similar to the method of Kim et al. (2006). Feature selection involved manual intervention: 83 "discriminating words" were selected that were present in 260 relevant documents from the DIP. These words explicitly excluded gene or protein names, but had a frequency in relevant abstracts that was highly improbable under the Poisson distribution for the word derived from 65,807 Medline abstracts mentioning *Saccharomyces cerevisiae*. Poisson parameters for each term in relevant/irrelevant documents were estimated from the 260 yeast PPI examples and the 65,807 background abstracts respectively, and the trained classifier calculated scores on another set of 325 abstracts (70 relevant, 255 irrelevant).

The evaluation did not quote standard performance metrics, but it can be inferred from one of the figures that the precision in the test was 77% at 55% recall. The DIP classifier (Marcotte et al. 2001) was later implemented as a retrieval service (Goetz and von der Lieth 2005).

2.5.3 Extension of literature collections

Besides finding relevant articles for database curators, filtering Medline using supervised learning has been applied to other resources that collect articles on a family of topics. The ACP Journal Club re-publishes literature for use in evidence-based medicine, with four sub-tasks of identifying articles relevant to therapy, diagnosis, etiology and prognosis. One study compared classifiers at detecting articles on each topic (Aphinyanaphongs et al. 2005). Articles featured in the ACP over a period of two years formed the relevant training corpus, with irrelevant examples being the remaining articles in journals that the ACP reviews. The performance comparison was based on ROC area macro-averaged under 5-fold cross validation, and tuning SVM parameters was done by the hold-out method (a 70/30 train and hold-out split inside the training data). Text pre-processing included stopword removal, case-folding and Porter stemming, and use of "whole" MeSH terms (descriptor plus qualifiers) as discussed in Section 2.4.2. In the comparison, the tuned SVMs outperformed a naïve Bayes method (precise form of the latter was not specified), as well as PubMed queries developed for each of the four categories. Another study (Bartling et al. 2003) filtered Medline records for those about dental research using Identify Patient Sets (Aronis et al. 1999).

2.5.4 Related articles for Medline filtering

Wilbur (2000) investigated the use of adaptive boosting (AdaBoost) and staging to classify documents for REBASE, a database of restriction enzymes. REBASE provided 3,121 relevant articles for the initial corpus, and vector cosine retrieval on the text of each abstract retrieved 200 "nearest neighbour" related abstracts to provide irrelevant Medline records (in the sense of "not found in REBASE"). The validation method was to average performance over 100 repeats of train-and-test. Boosting uses a compound classifier, which is a weighted sum of a sequence of classifiers. For each classifier in the sequence, documents

are given different error costs so as to choose a classifier that focuses on correctly classifying examples that were mis-classified by the previous one. They found that the precision averaged over the top 100 test documents increased significantly but not substantially with AdaBoost. The staging method trains one classifier (naïve Bayes), and uses it to score the "irrelevant but related" training documents. It then trains a second classifier (linear SVM) on the same relevant training documents and high-scoring irrelevant training documents. In the test phase, test documents that have high scores under the first classifier receive new scores that combine the score of the first and second classifier — which did substantially improve precision in the top 100 results.

PubMed related articles (Lin and Wilbur 2007) has also been used in Medline filtering. In what might be described as an inversion of k-Nearest Neighbours, the method of Liu and Altman (1998) took a bibliography of 87 articles on a highly specific topic, retrieved the top 40 related articles for each. It scored each distinct article by the number of related article lists in which it appeared, weighting each appearance by its rank out of 40. A "similar method" ranks the relatives of a set of abstracts in HubMed (Eaton 2006). Perez-Iratxeta et al. (2003) present an ad hoc method to rank the aggregated relatives of articles in a collection based on their word content. The above methods use related articles to get a set of close relatives for each document in the input set, and re-rank the relatives to produce a result set.

2.6 Classifying all of Medline

This section examines studies that classified or considered classifying Medline or a large representative subset of Medline (such as contiguous years). In relation to those studies it then discusses the issue of evaluating a classifier on a large subset of Medline, and the efficiency of Medline classifiers.

2.6.1 Related work on classifying all of Medline

The database curation filters for PreBIND (Donaldson et al. 2003) and PharmGKB (Rubin et al. 2005), already discussed, evaluated classifiers on small enriched subsets of Medline.

Unlike the IEDB (Wang et al. 2007), they did not already have a subset of Medline containing most relevant documents to use as the classification domain, and so turned to classifying larger subsets of Medline: a subset of 1.88 million records in Donaldson et al. (2003), and all of Medline in Rubin et al. (2005).

Suomela and Andrade (2005) presents an ad hoc linear classifier, with the score of a feature being the ratio of its frequencies in relevant and irrelevant documents, and the score of a document being its average feature score. The classifier was trained on 81,416 relevant documents having MeSH terms relating to stem cells and an equal number of random Medline records for irrelevant documents. The trained classifier obtained a precision-recall break-even of 0.65 on a test corpus of 6,923 random documents of which 204 had been manually judged relevant (2.9% prevalence). It was intended for the classifier to eventually be applied to all of Medline.

PubFinder (Goetz and von der Lieth 2005), was intended to classify large parts of Medline, consisting of one or more consecutive years. It re-implemented the method of Marcotte et al. (2001) behind a Web-service, but made no evaluation of effectiveness, and the evaluation in Marcotte et al. (2001) on yeast abstracts was itself unconventional. PubFinder used 100 title/abstract words having improbable frequencies in the examples under each word's Poisson distribution derived from Medline since 1990. It was reported to classify recent years of Medline at a rate of 500,000 abstracts per 1–3 minutes, but appears unmaintained, with queries submitted in mid-2006 still processing, and the counter of jobs completed from the queue incrementing every 2–3 days.

2.6.2 Evaluation of classifiers on all of Medline

When evaluating a classifier for the purpose of predicting its performance in operation, the documents on which the classifier is tested should be a representative sample of the documents which the classifier is expected to encounter in operation. In the case of the IEDB classifier (Wang et al. 2007), for example, cross validation is performed on an initial corpus which is a subset of Medline produced by a pre-filter query. The classifier will then perform similarly on future results of the pre-filter query. However, the study indicates that the classifier would also be used to find documents on sub-topics which were not represented in the pre-filter. It demonstrated moderately reduced performance in train-

and-test scenarios where test documents were from a sub-topic not included in the training set, because the classifier only learned a portion of the discriminating terms from the training data.

Classifiers also have reduced performance when the prevalence of relevant documents is lower, in particular causing lower precision for a given false positive rate. Donaldson et al. (2003), for example, obtained cross validation precision values on an enriched initial corpus, and intended to then apply the trained classifier to an operational data set of 1.88 million Medline records expecting 92% precision as in testing. However, lower prevalence in the large Medline sample means less precision should be expected. One might consider taking a random sample of Medline and manually classifying it to create cross-validation corpus. However, the prevalence of a particular topic in Medline is so low (typically 0.1% or less), that only a handful of relevant documents would occur out of tens of thousands. Therefore, enriched corpora are necessary for cross-validation experiments, which can then be used for comparing classifiers, but should not be taken to predict performance on Medline as a whole, where relevant documents are much rarer.

To test on a representative subset of Medline, one option is to manually evaluate the results of filtering. This was done for the PharmGKB classifier in Rubin et al. (2005), where a pharmacologist judged precision to be high in the top 1,649 results of the classifier having gene-drug co-occurrences. The test, however, requires an expert in the subject domain to judge relevance, does not measure recall, and has a subjective element. As an alternative, Chapter 3 will present a relative retrieval test which can evaluate performance at retrieval of a set of known relevant documents from a large Medline sample, although this is necessarily an under-estimate since some unknown relevant documents will also be retrieved and counted as irrelevant.

Besides the issue of test documents not being representative of Medline, most classifiers in the previous section were also trained on enriched initial corpora, where relevant documents are not only more prevalent but have a greater degree of similarity to irrelevant documents. This may not optimally train the classifier for distinguishing relevant documents from Medline background. In Chapter 3 we use all of Medline as unlabelled (but approximately irrelevant) data in the learning step before ranking Medline, which may be more effective.

2.6.3 Classifier efficiency

The efficiency of a classifier is the speed at which it operates, in terms of documents classified per second. Comparing the efficiency of classifiers on a given task is not as objective as comparing effectiveness, because efficiency is partly due to factors other than the choice of classifier, including the hardware and the method of implementation. With choice of classifier, some classifier algorithms are inherently more efficient than others. For example, all linear or log-linear classifiers (naïve Bayes, perceptron, SVM) have the same algorithmic complexity when classifying, so differences in speed depend only on hardware and implementation factors. Training steps of these classifiers, although not strictly "speed of classification" also contribute to the total time to complete a classification task. For example, Nearest Neighbours needs no training at all, while Naïve Bayes requires time proportional to the number of training documents, and SVMs, neural networks and boosting algorithms that require iteration or optimisation can take much longer for large training sets. The effect of hardware is simply that faster computers take less time to perform a given classification task, and some algorithms are also faster with additional memory. With the implementation itself, there are many ways of making a classifier faster by using special data structures or using a different programming language for the most computationally intensive steps.

Efficiency is mentioned briefly, if at all, in Medline classification studies because it is often not important for the application. For example, a small domain of operation requires less efficiency than classifying all of Medline. In Wang et al. (2007), the rate of 1,000 Medline records per 30 seconds was acceptable, because the domain constructed by the IEDB's pre-filter query only contained around 20,000 documents. Also, if a classification task only needs to be performed once, it is not worth the effort of making a fast implementation, which is why the predicted 3 days to calculate scores for all of Medline was acceptable in Donaldson et al. (2003). Efficiency, however, becomes important when the domain is large (such as classifying all of Medline) and the task has to be performed repeatedly, especially over a Web interface. Speed of training also becomes important when more than a few thousand training documents are involved. Chapter 3 will present several methods for vastly speeding up the training and execution of a simple Bayes classifier in order to make whole-Medline classification usable from a Web front-end.

Chapter 3

Methods

This chapter formulates a binary naïve Bayes classifier using the multivariate Bernoulli document model, and presents options for estimating its parameters under conditions of extreme class skew. These are followed with the methods for indexing Medline record fields, criteria for feature selection, and how the XML distribution of Medline is pre-processed to allow classification of all of Medline in a short period of time. Other processes described here include the operation of filtering all of Medline, the structure and functions of the Web front-end, and the cross validation process for evaluating the classifier — which is also available over the Web front-end. With the description of the underlying methods complete, we present the data sets and experiments for Chapter 4. These experiments are designed to identify the overall best variant of the classifier across a range of tasks, analyse the final classifier's performance characteristics, and compare it to a related classifier-filter of Medline records.

3.1 Classifier formulation

To filter Medline for relevant records, we develop the binary classification version of naïve Bayes with multivariate Bernoulli document model (Section 2.2.3), which is also known as the Binary Independence Model (Lewis 1998). We introduce modifications to the Laplace method of smoothing parameter estimates to counteract bias when the training data is highly skewed.

3.1.1 Naïve Bayes formulation

The naïve Bayes classifier predicts the maximum a posteriori (MAP) hypothesis, which is the most probable class for the document under the model parameters. For a binary classifier, r and $\tilde{r}$ represent the complementary classes of "relevant" and "irrelevant", and the set of classes is $\mathcal{C} = \{r, \tilde{r}\}$. Inside a probability function, r and $\tilde{r}$ represent the events $C = r$ and $C = \tilde{r}$ where the Bernoulli random variable C for the class generates a relevant or irrelevant document label. The MAP hypothesis in Equation 2.2 then reduces to $c_{MAP} = r \iff P(r|d) > P(\tilde{r}|d)$ for some document d. This condition, which predicts relevance when it is more probable than irrelevance, is equivalent to the condition $S(d) > 0$, where the classifier score $S(d)$ is the posterior log odds of relevance. Below we use Bayes rule to express $S(d)$ in terms of the likelihood ratio and prior odds of relevance:

$$\begin{aligned} S(d) &= \log \frac{P(r|d)}{P(\tilde{r}|d)} \\ &= \log \frac{\frac{P(r)P(d|r)}{P(d)}}{\frac{P(\tilde{r})P(d|\tilde{r})}{P(d)}} \\ &= \log \frac{P(d|r)}{P(d|\tilde{r})} + \log \frac{P(r)}{P(\tilde{r})} \end{aligned} \tag{3.1}$$

The indexing process further represents the document d by a feature vector $\mathbf{f} = (f_1, f_2, \ldots, f_{|\mathcal{V}|})$, with each $f_i \in \{0, 1\}$, representing presence (1) or absence (0) of vocabulary term i. Inside a probability expression, f_i represents the event $F_i = f_i$ where the random variable F_i generates an occurrence or non-occurrence. Assuming independence amongst the features conditional on the class (Equation 2.3) yields a classifier score of

$$\begin{aligned} S(\mathbf{f}) &= \log \prod_{i=1}^{|\mathcal{V}|} \frac{P(f_i|r)}{P(f_i|\tilde{r})} + \log \frac{P(r)}{P(\tilde{r})} \\ &= \sum_{i=1}^{|\mathcal{V}|} \log \frac{P(f_i|r)}{P(f_i|\tilde{r})} + \log \frac{P(r)}{P(\tilde{r})} \end{aligned} \tag{3.2}$$

We now want to express the classifier score $S(\mathbf{f})$ in terms of model parameters, which can be estimated. When the statistical model generates a document, the Bernoulli variable C first generates the class label c (either r or $\tilde{r}$). c in turn specifies the mixture component $\mathbf{F}|C = c$, which is a multivariate Bernoulli distribution to generate a feature vector $\mathbf{f}$ of the specified class. The multivariate Bernoulli distribution consists of $|\mathcal{V}|$ independent Bernoulli random variables, $F_i|C = c$, each of which has parameter θ_{ci}, which is the probability of feature i occurring in a document of class c. Using the fact that $f_i \in \{0, 1\}$, we write each feature probability in Equation 3.2 in the form

$$P(f_i|c) = \theta_{ci}^{f_i}(1 - \theta_{ci})^{1-f_i} = \begin{cases} \theta_{ci} & \text{if } f_i = 1 \\ 1 - \theta_{ci} & \text{if } f_i = 0 \end{cases} \tag{3.3}$$

Because only a tiny fraction of the vocabulary terms will be present in any particular document, $S(d)$ can be calculated much faster by expressing it as a base score for a document where all $f_i = 0$, and a sum over each feature that occurs. Substituting Equation 3.3 into Equation 3.2 we obtain (Lewis 1998):

$$S(\mathbf{f}) = \sum_{i=1}^{|\mathcal{V}|} f_i \left(\log \frac{\theta_{ri}/(1 - \theta_{ri})}{\theta_{\tilde{r}i}/(1 - \theta_{\tilde{r}i})} \right) + \sum_{i=1}^{|\mathcal{V}|} \log \frac{1 - \theta_{ri}}{1 - \theta_{\tilde{r}i}} + \log \frac{\theta_r}{1 - \theta_r} \tag{3.4}$$

Where θ_r is the Bernoulli parameter for the class variable C, and is equal to the prior probability $P(r)$ of the document class being relevant. We define the feature weights w_i in Equation 2.1 to be the log odds ratio for the feature occurring in relevant vs irrelevant documents:

$$w_i = \log \frac{\theta_{ri}/(1 - \theta_{ri})}{\theta_{\tilde{r}i}/(1 - \theta_{\tilde{r}i})} \tag{3.5}$$

Using b for the constant terms in Equation 3.4 yields the form of a linear classifier:

$$S(\mathbf{f}) = \sum_{i=1}^{|\mathcal{V}|} w_i f_i + b = \mathbf{w} \cdot \mathbf{f} + b \tag{3.6}$$

3.1.2 Parameter estimation

Training the classifier consists of estimating its Bernoulli parameters: θ_r for the class random variable C, and θ_{ci} for the random variable $F_i|C = c$ which models the occurrence or non-occurrence of each feature in a given class. We estimate θ_r using Laplace smoothing on the number of relevant (N_r) and irrelevant ($N_{\tilde{r}}$) documents in the training corpus. The total number of training documents is $N = N_r + N_{\tilde{r}}$:

$$\hat{\theta}_r = \hat{P}(r) = \frac{N_r + 1}{N + 2} \tag{3.7}$$

To estimate θ_{ci} we start with the MAP estimate of Equation 2.4 using a beta distribution prior for Bayesian smoothing. N_{ci} is the number of occurrences of term i in documents of class c and N_c is the number of documents in class c. We assume binary features, where each term occurs exactly 0 or 1 times in a given document. The beta distribution parameters a_{ci} and b_{ci} have subscripts to indicate there may be a different prior on each θ_{ci}:

$$\hat{\theta}_{ci} = \hat{P}(F_i = 1|C = c) = \frac{N_{ci} + a_{ci}}{N_c + a_{ci} + b_{ci}} \tag{3.8}$$

The maximum likelihood estimate (no smoothing), would set $a_{ci} = b_{ci} = 0$. As discussed in Section 2.2.4, the estimate is biased, and can produce infinite feature weights w_i when N_{ci} is zero — that is, for features that fail to occur in the training documents of one class. To avoid this, Laplace smoothing is commonly used in cross validation on balanced initial corpora. Laplace smoothing sets the beta distribution parameters to $a_{ci} = b_{ci} = 1$, resulting in a flat prior distribution on θ_{ci}, which in the absence of evidence ($N_{ci} = N_c = 0$) produces a prior estimate of $\theta_{ci} = 0.5$. However, with a skewed class distribution ($N_r \ll N_{\tilde{r}}$), these counts affect the estimate of θ_{ri} much more than the estimate of $\theta\tilde{r}i$, biasing the value of w_i upwards. The smoothing most strongly affects rare features, producing large positive values of w_i even for features that are more frequent in irrelevant documents, biasing the classifier in favour of the relevant class and reducing classifier performance. We consider two smoothing methods based on pseudocounts to correct the bias of Laplace smoothing under skewed class distributions.

The previous version of the classifier in Poulter et al. (2008) used an approach called Background smoothing, which sets $a_{ci} = N_i/N$, being the background frequency of term i across both classes in the training data, and $a_{ci} + b_{ci} = 1$ in the denominator, to contributes a total of one article worth of evidence. The small hyperparameters provide weak smoothing to prevent rare features from obtaining positive weights when they fail to occur in any relevant articles, thus reducing the bias toward relevant articles under a skewed class distribution. The prior prior distribution on θ_{ci} is skewed to-wards zero, making prior estimate of θ_{ci} equal to a_i (the global frequency of feature i) instead of 0.5. However, we later found that the weak smoothing permits more extreme feature weights in both directions when $N_{ci} = 0$, harming performance in some cases.

In this version of the classifier we propose an original modification to Laplace smoothing of a beta distribution prior, which we call split-Laplace smoothing, and uses the pseudocount interpretation of a_i and b_i. It divides the Laplace pseudocounts between the two classes according to the prior probability of each class to counteract class skew. For each term i, ordinary Laplace smoothing introduces 2 fictitious documents in each class: one with term i and one without. Split-Laplace smoothing introduces 4 fictitious documents of unknown class: 2 with term i and 2 without. The values a_{ci} and b_{ci} are then the expected number of (non-)occurrences in class c according to the prior probability $P(c)$, yielding $a_{ri} = b_{ri} = 2\theta_r$ and $a_{\tilde{r}i} = b_{\tilde{r}i} = 2(1-\theta_r)$. Split-Laplace reduces to Laplace smoothing when $\theta_r = 0.5$ (that is, perfectly balanced data), and like Laplace smoothing estimates $\hat{\theta}_{ci} = 0.5$ in the absence of any evidence. As the training data becomes more skewed/unbalanced ($P(r) \ll P(\tilde{r})$), split-Laplace smoothing starts to skew the prior distribution on θ_{ri} toward 0/1 and the prior on $\theta_{\tilde{r}i}$ toward 0.5, with weaker smoothing on θ_{ri} than on $\theta_{\tilde{r}i}$. The adapted pseudocounts almost entirely counteract class skew, so that $N_{ri} = N_{\tilde{r}i} = 0$ produces only a small positive weight feature weight instead of the large $\log \frac{N_{\tilde{r}}+1}{N_r+1}$ weight that Laplace smoothing would have produced.

3.2 Medline record indexing

This section describes the features that the indexing step extracts from Medline records, and how Medline is pre-processed so that all 16 million records can be classified in a short period of time.

3.2.1 Medline record fields

In the Baseline (NLM 2007b) distribution of Medline, the records are provided in an XML format, an example of which is shown in Figure 3.1, from which features can be extracted. We consider the following sources of features in Medline records:

- Word: Each feature is a token from the `<ArticleTitle>` or `<AbstractText>` field, extracted using the text indexing algorithm below.
- MeSH: The Medical Subject Heading descriptor phrases from the `<DescriptorName>` fields of the XML.
- Qual: The MeSH qualifier phrases from the `<QualifierName>` fields of the XML (there are only about 90 qualifiers in use).
- ISSN: The ISSN of the journal, such as "0147-5185", from the `<ISSN>` field of the XML.
- Author: The initials and last name of authors, for example "JH Bloggs", from the `<Initials>` and `<LastName>` field of each `<Author>` field.

Each feature is a string (a sequence of characters), which is associated with a feature type (Word, MeSH, Qual, ISSN or Author). We keep the feature types separate: "mouse" as a MeSH term has a different feature ID from "mouse" as a title/abstract Word. All of the feature spaces together produce the complete WMQIA feature space. In Poulter et al. (2008), the classifier used a much smaller "MQI" feature space composed of only MeSH, Qual and ISSN features.

Text indexing (Section 2.4.1) is used to extract features from the title/abstract to produce the Word feature space. The indexing algorithm below does not use Porter stemming, attempts to maintain hyphenated terms such as "A*0201-restricted", uses partial case-folding, and discards numbers and punctuation outside of words:

- Concatenate title and abstract, adding a padding space to separate them.
- Split the text on spaces and certain non-word characters, by finding matches to the Python regular expression `\s|\'s|--|[!/<>=#$^\"{};:&]`, thus yielding raw terms.

```
<PubmedArticle>
   <MedlineCitation Owner="NLM" Status="MEDLINE">
      <PMID>8372948</PMID>
      <DateCreated><Year>1993</Year><Month>10</Month><Day>12</Day></DateCreated>
      <DateCompleted><Year>1993</Year><Month>10</Month><Day>12</Day></DateCompleted>
      <DateRevised><Year>2004</Year><Month>11</Month><Day>17</Day></DateRevised>
      <Article PubModel="Print">
         <Journal>
            <ISSN IssnType="Print">0147-5185</ISSN>
            <JournalIssue CitedMedium="Print">
               <Volume>17</Volume><Issue>10</Issue>
               <PubDate><Year>1993</Year><Month>Oct</Month></PubDate>
            </JournalIssue>
            <Title>The American journal of surgical pathology</Title>
            <ISOAbbreviation>Am. J. Surg. Pathol.</ISOAbbreviation>
         </Journal>
         <ArticleTitle>Primary angiosarcoma of the spleen. A clinicopathologic study of 40 cases.</ArticleTitle>
         <Abstract><AbstractText>Forty primary splenic[...]</AbstractText></Abstract>
         <Affiliation>Department of Hematologic and Lymphatic Pathology[...]</Affiliation>
         <AuthorList CompleteYN="Y">
            <Author ValidYN="Y"><LastName>Falk</LastName><ForeName>S</ForeName><Initials>S</Initials></Author>
            <Author ValidYN="Y"><LastName>Krishnan</LastName><ForeName>J</ForeName><Initials>J</Initials></Author>
            <Author ValidYN="Y"><LastName>Meis</LastName><ForeName>J M</ForeName><Initials>JM</Initials></Author>
         </AuthorList>
         <Language>eng</Language>
         <PublicationTypeList><PublicationType>Journal Article</PublicationType></PublicationTypeList>
      </Article>
      <MedlineJournalInfo>
         <Country>UNITED STATES</Country><MedlineTA>Am J Surg Pathol</MedlineTA><NlmUniqueID>7707904</NlmUniqueID>
      </MedlineJournalInfo>
      <ChemicalList>
         <Chemical>
            <RegistryNumber>0</RegistryNumber><NameOfSubstance>Biological Markers</NameOfSubstance>
         </Chemical>
      </ChemicalList>
      <CitationSubset>IM</CitationSubset>
      <MeshHeadingList>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Adult</DescriptorName>
         </MeshHeading>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Aged</DescriptorName>
         </MeshHeading>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Aged, 80 and over</DescriptorName>
         </MeshHeading>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Biological Markers</DescriptorName>
            <QualifierName MajorTopicYN="N">analysis</QualifierName>
         </MeshHeading>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Female</DescriptorName>
         </MeshHeading>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Hemangiosarcoma</DescriptorName>
            <QualifierName MajorTopicYN="N">chemistry</QualifierName>
            <QualifierName MajorTopicYN="Y">pathology</QualifierName>
            <QualifierName MajorTopicYN="N">physiopathology</QualifierName>
         </MeshHeading>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Humans</DescriptorName>
         </MeshHeading>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Immunohistochemistry</DescriptorName>
         </MeshHeading>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Male</DescriptorName>
         </MeshHeading>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Middle Aged</DescriptorName>
         </MeshHeading>
         <MeshHeading>
            <DescriptorName MajorTopicYN="N">Splenic Neoplasms</DescriptorName>
            <QualifierName MajorTopicYN="N">chemistry</QualifierName>
            <QualifierName MajorTopicYN="Y">pathology</QualifierName>
            <QualifierName MajorTopicYN="N">physiopathology</QualifierName>
         </MeshHeading>
      </MeshHeadingList>
   </MedlineCitation>
   <PubmedData>
      <History>
         <PubMedPubDate PubStatus="pubmed"><Year>1993</Year><Month>10</Month><Day>1</Day></PubMedPubDate>
         <PubMedPubDate PubStatus="medline">
            <Year>1993</Year><Month>10</Month><Day>1</Day><Hour>0</Hour><Minute>1</Minute>
         </PubMedPubDate>
      </History>
      <PublicationStatus>ppublish</PublicationStatus>
      <ArticleIdList><ArticleId IdType="pubmed">8372948</ArticleId></ArticleIdList>
   </PubmedData>
</PubmedArticle>
```


Figure 3.1: Medline XML for PubMed ID 8372948, edited for space. See Figure 2.3 for the older MEDLINE format.

- Strip any non-alphanumeric characters (neither letters nor numbers) from the beginning and end of each term. This discards parentheses on the edge of expressions but avoids splitting up expressions such as "I-A(b)-restricted".
- Discard the term if it looks like a number or otherwise contains no alphabetic characters by finding matches to the regular expression `^[0-9eE\W]+$`.
- Discard the term if its lower case version is found in the stopword list on `http://www.lextek.com/manuals/onix/stopwords2.html` (this includes single-letter terms).
- If the first character of the term is a letter, make that letter upper-case. All words are therefore spelled as if they had occurred the beginning of a sentence. This obtains the main benefit of case-folding, but avoids, for example, conflating human and mouse gene names (PON3 and Pon3), which often differ only in case.

3.2.2 Pre-processing steps and data structures

We first describe a simple, but inefficient, implementation of a naïve Bayes text classifier that does not use the pre-processing steps or data structures mentioned later. The simple classifier would represent documents as a list of feature strings, since the feature vector $\mathbf{f}$ is sparse, consisting mostly of zeros. For training, it would read the training corpus from disk, extract string features from each document as necessary, count the occurrences of features in relevant and irrelevant documents, and calculate the weights w_i for each feature string. For testing or operation, it would read in the full text (or XML) of each test document in turn, extract feature strings, use Equation 3.6 to calculate the score of each test document by summing the associated feature weights, and print the IDs of documents having positive scores. Because this simple classifier operates directly on the text and feature strings of documents, it would take up to a week to process Medline consisting of 80GB of XML (uncompressed), use over a gigabyte of memory to keep a lookup table for 12 million features weights, and take an additional week of parsing if the "rest of Medline" were used in training for the irrelevant documents.

Fast Medline classification comes at the cost of the complexity of maintaining the following data structures. First, the program maintains a database for looking up documents by PubMed ID, so that it can accept training documents as PubMed IDs instead of as full

text. Next, because feature extraction is expensive, it is only performed once for each record, with the resulting feature vectors stored in a database. Because feature strings take up a lot of space, the feature vectors are also compacted by mapping the feature strings onto numerical feature IDs, which in turn requires maintaining another database to translate between feature strings and their IDs. However, even looking up compact feature vectors from a database is a slow operation, so the vectors are added in parallel to a flat file which can be processed quickly when the classifier is calculating scores for every record in Medline. Lastly, to be able to quickly use the "rest of Medline" as training data, the occurrence count in Medline of every feature is tracked during the feature extraction step. These data structures also need to be updated in parallel as new records are added to Medline each day.

The process of updating these data structures, whether from daily Medline updates or when parsing the Baseline XML (NLM 2007b) distribution of Medline at the beginning of the year, is depicted in the data flow diagram of Figure 3.2. The major steps in the process are:

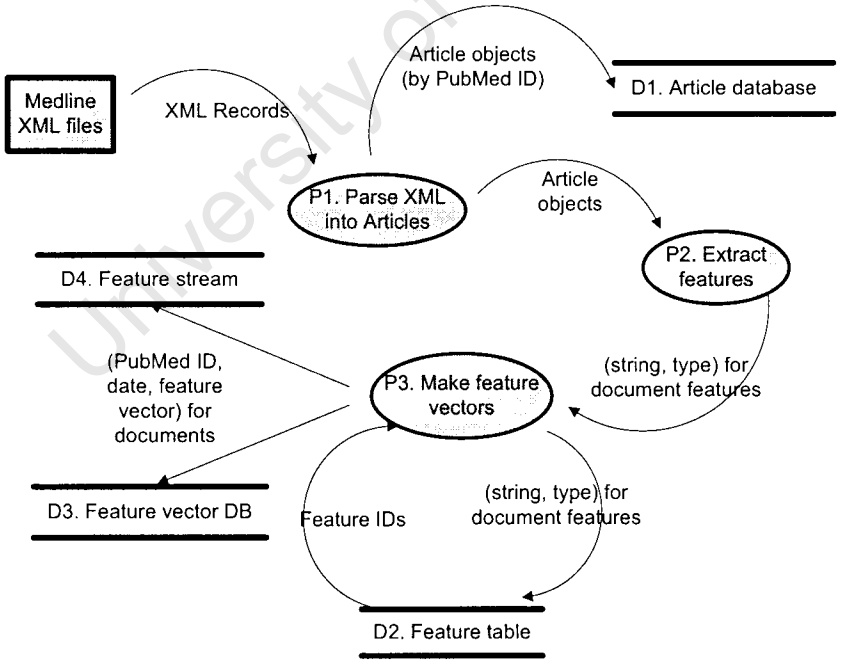


Figure 3.2: Data flow diagram for pre-processing each XML file in the Medline Baseline distribution (NLM 2007b). The process steps are P1-P3, data stores are D1-D4, and directional arcs indicate the data passing between them.

- P1: The parsing step processes XML files using cElementTree (ElementTree 2008) and maintains a list of already-processed files. Each file such as `medline08n0479.xml.gz` contains around 30,000 Medline records like the one in Figure 3.1. Parsing produces Article objects, having fields for publication date, title, MeSH terms and so forth.

- P2: The feature extraction step converts Articles into a list of feature strings. For each Article object, call the appropriate method to convert the record fields into a data structure that for each feature type (MeSH, Word, etc.) lists the feature strings of that type that occurred in the article.

- P3: The feature mapping step converts the list of feature strings into a list of feature IDs. This requires mapping each unique pair of feature string and feature type to a feature ID. Each feature ID is also an index i in the mathematical form of the feature vector. This step also tracks the number of occurrences of each feature ID in Medline for using the "rest of Medline" in training.

The on-disk data structures are as follows:

- D1: The Article database is a Berkeley DB (Oracle 2008) which provides lookup of Article objects by PubMed ID. These objects are used to format the result pages, and to regenerate the other databases without re-parsing Medline. Berkeley DB was chosen over the SQLite relational database because for simple object insertion and retrieval it is more efficient than an indexed SQLite table.

- D2: The feature table is an SQLite (SQLite 2008) database table having fields for feature ID, feature string, feature type, and occurrence count. The feature ID is also the row number, which increments when new features are added. The feature string is the raw feature as encountered in the text (such as "Pharmacology") and the feature type distinguishes the source of the feature (say, "Word" or "Qual"). The occurrence count is the number of times the feature was encountered during parsing of Medline. We maintain two interfaces to the table: a fast one that works with in-memory data structures and writes to disk in one pass when the program ends, and a slow one that makes database calls for every operation (see Section 5.3).

- D3: The database of feature vectors is an SQLite table with fields for PubMed ID, date on which Medline record was completed, and feature vector. It is used to look

up the feature vectors for user-submitted PubMed IDs, and additionally in cross validation to retrieve feature vectors for a random sample of PubMed IDs.

- D4: The stream of feature vectors is a compact binary file listing PubMed ID, date and feature vector for each Medline record. It is redundant to the database, but is several times faster to iterate over when filtering Medline.

Before being stored in the database or stream of feature vectors, the lists of feature IDs are further compressed using variable byte encoding (Manning et al. 2008; Chapter 5). This produces space savings for the feature stream presented in Section 4.5. Variable byte encoding sorts the list by increasing feature ID and calculates the difference or gaps between successive 32-bit IDs. These gaps are much smaller than the IDs themselves so most of the 32 bits go unused, a property that the encoding uses to write each gap with a variable number of bytes. Each byte stores 7 bits of the gap with the 8th bit set to 0, except in the last byte where the 8th bit is set to 1 to signal the end of the number. Small gaps require one or two bytes instead of four, and the feature vector is written as a sequence of gaps between successive feature IDs. When the feature stream is highly compressible and reading it from disk dominates the running time, variable byte encoding greatly speeds up classification.

3.2.3 Feature selection

We use feature selection, discussed in Section 2.4.3, to remove features that contain little information about the class variable, and are thus candidates for overfitting or introducing noise into the document scores. Features are selected based on their occurrence counts in training data, the same counts that are used to estimate the parameters in Equation 3.8. In cross validation, this means that slightly different features are selected in each fold, because only 90% of the initial corpus is provided as training data. If the whole initial corpus were used to select a single set of features for all validation folds, it would risk some overfitting to test data (Section 2.3.2).

For the document frequency method of selecting features, where $N_i = N_{ri} + N_{\bar{r}i}$, we evaluate thresholds on N_i of 0, 1, 2, 3, 4 and 8. $N_i \geq 0$ selects all features, because out-of-vocabulary (OOV) features having $N_i = 0$ in the training data may still occur in test data

and so contribute to classification. Out-of-vocabulary features are excluded by requiring $N_i \geq 1$. $N_i \geq 2$ or $N_i \geq 3$ may additionally remove spelling errors and nonsense strings, although informative words may also occur once or twice if there is a lack of training data. The criteria of $N_i \geq 4$ and $N_i \geq 8$ further eliminate words where there is little information available to estimate its Bernoulli parameters in relevant/irrelevant articles. Again, this criterion may exclude informative features if there are too few training examples.

Wang et al. (2007) used the information gain criterion from Equation 2.6 to select features, selecting feature i only if it meets the criterion $IG(C, F_i) > 2 \times 10^{-5}$. The IEDB initial corpus was balanced, but if the initial corpus is highly skewed the random variable C already will have very low entropy, so fewer bits can be gleaned from partitioning on each feature. Extreme class skew is a fact of Medline, but is not reflected in existing Medline corpora for supervised text classification. The general approach of selecting a number of features as in Zheng et al. (2004) is not suitable here, because of the requirement for a tuning step to determine the optimal number of features. Instead we opt to account for class skew in the feature selection criterion itself, by dividing information gain (reduction in entropy of the class variable) by the original entropy of the class variable. This was introduced as gain ratio in the context of decision tree classifiers by Quinlan (1986):

$$GR(C, F_i) = \frac{IG(C, F_i)}{H(C)} = \frac{H(C) - H(C|F_i)}{H(C)} \tag{3.9}$$

In the case of perfectly balanced training data ($P(r) = P(\tilde{r}) = 0.5$), the entropy of the class variable $H(C)$ is 1, so the gain ratio reduces to ordinary information gain. In Section 3.4.3 we describe experiments to determine a single threshold on gain ratio that works well under a wide range of input conditions.

To calculate gain ratio, we use estimates of $P(c, f_i)$, $P(c)$ and $P(f_i)$ for Equation 2.6. We do not used smoothed estimates, because the calculation does not suffer from infinities as in the calculation of w_i. $P(c, f_i) = 0$ only makes one component of the expression zero, and $P(F_i = f_i) = 0$ just results in $P(r, f_i) = 0$ and $P(\tilde{r}, f_i) = 0$. The estimated probabilities are:

$$
\begin{aligned}
\hat{P}(C = c) &= \frac{N_c}{N} \\
\hat{P}(C = c, F_i = 1) &= \frac{N_{ci}}{N} \\
\hat{P}(C = c, F_i = 0) &= \frac{N_c - N_{ci}}{N} \\
\hat{P}(F_i = 1) &= \frac{N_i}{N} \\
\hat{P}(F_i = 0) &= \frac{N - N_i}{N}
\end{aligned} \tag{3.10}
$$

As stated previously, feature selection is performed as part of the classifier training step. We also use another process, called "vacuuming", not as part of training but as a speed-increasing transformation of the Medline data structures created by the pre-processing steps of Section 3.2.2. Vacuuming deletes entries in the feature table that occur only once or twice in all 16 million articles in Medline, then edits the feature vectors to make it as if those features never existed, and also freezes the vocabulary of Medline until the following year's release of a new Baseline distribution. Vacuuming reduces the size of the WMQIA vocabulary in Medline from around 12 million features to just under 3 million and speeds up the training step, which involves creating and manipulating dozens of $|\mathcal{V}|$-dimensional vectors. Vacuuming has negligible effect on performance in cross validation, because the vacuumed features are rare enough that the vast majority would be excluded even by an $N_i \geq 1$ criterion in each validation fold, as well as failing to occur in any of the test documents.

3.3 The classifier in operation

The implementation of the classifier, named MScanner, is written in the Python programming language (van Rossum and Drake 2001). We divided the program into two parts: a back-end library that performs Medline filtering, cross validation and performance evaluation, and updating of the database (Section 3.2.2) — and a front-end Web application and queue process that respectively interact with the user and control the execution of tasks and updates. In Section 4.4 we look at the presentation of the Web front-end from a user's

perspective.

3.3.1 Process for filtering Medline

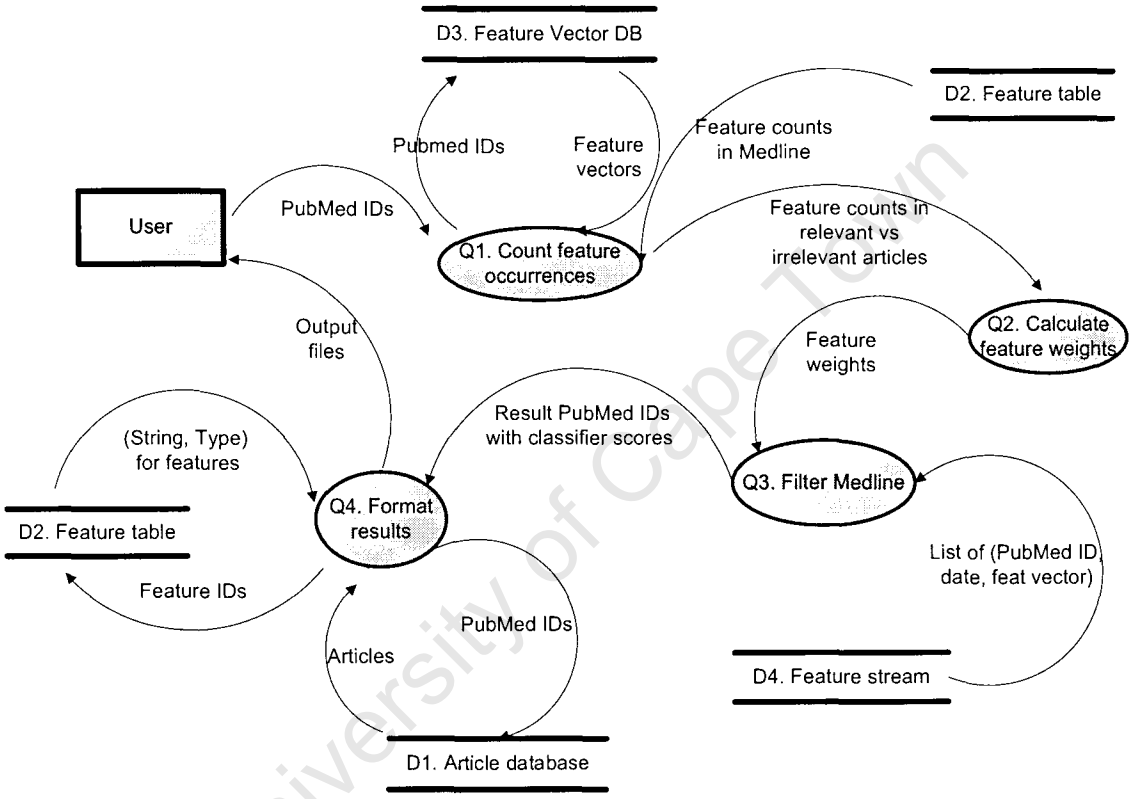


Figure 3.3: Data flow diagram for the process of filtering Medline. The input is a list of PubMed IDs representing relevant training examples, and the output consists of HTML files with formatted results.

The back-end of the classifier is a Python library for performing Medline filtering, and cross validation tests. Figure 3.3 depicts, at a high level, the data flow from the perspective of the back-end code. The following steps are performed in the process of filtering Medline:

- Q1: Count feature occurrences. Receive the list of N_r relevant PubMed IDs, and retrieve the corresponding feature vectors from the database. Sum the feature vectors to obtain a $|\mathcal{V}|$-dimensional vector with the N_{ri} counts. From the feature table, obtain

the number of occurrences N_i of each feature in Medline (performed once at start-up). Then calculate term frequencies in irrelevant documents as $N_{\bar{r}i} = N_i - N_{ri}$. This is the rest-of-Medline approximation mentioned elsewhere, which uses the unlabelled records of Medline instead of user-specified irrelevant training examples.

- Q2: Calculate feature weights. Receive N_{ci} and N_c for relevant and irrelevant articles. Select features using the document frequency and gain ratio (Equation 3.9) criteria. For selected features, estimate the Bernoulli parameters using Equation 3.7 and Equation 3.8, and calculate feature weights w_i using Equation 3.5. Non-selected features have w_i set to zero.

- Q3: Filter Medline. Receive the feature weights w_i and constant term b, and calculate the classifier score $S(\mathbf{f})$ for all non-input Medline records using Equation 3.6. Apply limits on the minimum classifier score, sort the remaining PubMed IDs by decreasing score, apply the limit on the maximum number of results, and return the remaining PubMed IDs and associated scores. For maximum speed, Medline filtering is performed by a C program filtering the stream of feature vectors (feature stream is D4 in Figure 3.2).

- Q4: Format results. Retrieve Article objects for the result PubMed IDs, and format an HTML file with a table of result records, plus some additional files.

3.3.2 The Web front-end

The back-end is a library of code for performing Medline filtering and cross validation, and is used to carry out the experiments in Chapter 4. The front-end consists of a Web application written using the Web.Py (WebPy 2008) framework, and a long-running queue process that interacts with the back-end library. While this section describes the architecture of the front-end application, the results of Section 4.4 will look at it from a user perspective with an example of using the classifier to perform Medline filtering.

The web.py controller handles requests for five dynamic pages, each of which is generated using the Cheetah templating language (Cheetah 2008):

- Home: Introduction to the classifier and how to use it.

- Query: Form for submitting a task to the classifier.
- Status: Progress reports on the currently-running task and lists the queue of not-yet-started tasks.
- Output: List of non-hidden result sets, with a facility to delete old result sets.
- Contact: Web contact form to report problems.

When the user submits a task on the query page, the controller validates the form. If there are invalid parameters, the controller re-displays the query page with those fields marked. If all parameters are valid, the controller writes a *task descriptor file* in the queue directory, and redirects the user to the status page to monitor the progress of the query, which may be waiting, running, or complete. The status page also provides a link to the directory in which the result pages will be placed. The output page provides links to each set of results, in case the user went away without bookmarking the location of the result directory. It does not however list results for tasks where the "hide output" check box was ticked on the query form. The following parameters can be configured on the task submission page:

- The list of PubMed IDs (training examples of the class "relevant").
- Whether to filter Medline or perform cross validation.
- Whether to use "all features" (WMQIA) or just MQI features. There are fewer MQI features per document, allowing faster training and filtering.
- The name of the task, so that it can be identified later.
- A code that will be required in order to delete the output directory for the task (optional).
- The maximum number of records to return in Medline filtering.
- A minimum date. If provided, constrain Medline to contain only articles added since that date. This slows down training and filtering because whereas full-Medline counts are pre-calculated, the features in the date-constrained Medline have to be counted before they can be used in training. It is however useful for retrieving only new articles.

- The minimum score of a Medline record to return, changing the decision threshold from the naïve Bayes default of zero.

Because the controller has to respond immediately following submission of a query, it hands off the query to a queue process. The queue process is started separately, and runs indefinitely. On start-up it loads a list of all PubMed IDs in Medline and the vector N_i — Medline occurrence counts used for rest-of-Medline training. It then checks the queue directory once every second, reading the oldest task descriptor file present. The descriptor file contains a sanitised version of the input submitted to the query form. Having read a task descriptor, the queue process uses the back-end library to carry out the task. The library writes the output to a directory named after the task. Once a day, the queue process uses the back-end to process any new Medline update files (Figure 3.2), and delete the oldest outputs. Using a queue process also ensures that only one filtering task runs at a time, and that filtering does not take place during database updates.

3.4 Classifier evaluation

This section describes the methods for evaluating the classifier, beginning with the framework for cross validation experiments, and the initial corpora to be used in cross validation. It then presents two sets of experiments. The first set of experiments tests the performance of different methods considered in designing classifier, so that we can pick the best combination of methods. The second set of experiments analyses the performance characteristics of the resulting method, to understand why it performs as it does on different corpora. It represents the final iteration of the series of performance analysis experiments which guided the revision of the classifier into its present form. The remaining experiments compare the method to another classifier, and examines its efficiency. We also devise a "relative retrieval test" to test the classifier on a representative sample of Medline, because the cross validation experiments need a higher prevalence of relevant documents to provide sufficient training examples.

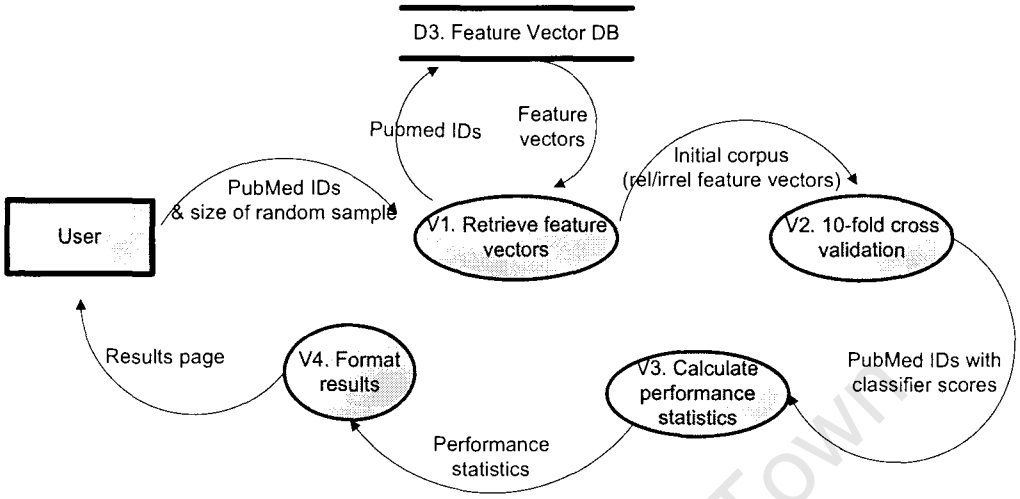


Figure 3.4: Data flow diagram for the process of cross validation described in Section 3.4.1. The input is a list of relevant PubMed IDs and the size of the Medline sample to use for irrelevant PubMed IDs. The output is and HTML file and images for the performance statistics.

3.4.1 Evaluation framework and performance statistics

The evaluation framework is 10-fold cross validation (Section 2.3.1), and the evaluation described below is provided as a function on the Web front-end as well as being used to obtain the results of Chapter 4. Each fold of cross validation presents different training data, so the entire training process of feature selection and parameter estimation is repeated in each of the 10 folds. The Web front-end also provides a facility for performing cross validation on new data sets, to evaluate the classifier in new conditions. Figure 3.4 depicts the main steps in validation.

- V1: Retrieve feature vectors. Receive a list of relevant training examples as PubMed IDs from the user, and either a list of specified irrelevant training examples or the number of PubMed IDs to randomly sample from Medline instead. In the latter case, the sample is taken from a pre-loaded vector of all PubMed IDs in Medline. Retrieve feature vectors for the PubMed IDs, forming the initial corpus of N_r relevant and $N_{\bar{r}}$ irrelevant documents.

- V2: 10-fold cross validation (as in Section 2.3.1). Shuffle the relevant and irrelevant subsets of the initial corpus, then divide each into 10 segments. In each round of

cross validation, 9 segments from each are used for training (feature selection and parameter estimation), and the trained classifier calculates scores for test documents in the remaining pair of segments. Repeating 10 times, with each pair of segments in turn forming the test corpus, yields classifier scores for all documents.

- V3: Calculate performance statistics. First micro-average the 10 test folds by aggregating them to reconstruct the initial corpus, with documents ordered by decreasing score. For each distinct threshold τ, calculate the (micro-averaged) confusion matrix at that threshold, consisting of TP, TN, FP and FN. This equals the sum of the confusion matrices that would be obtained within each validation fold. The result is a "vector of confusion matrices", equal to the number of distinct thresholds τ in the results. Use the information to construct TPR and FPR vectors for the ROC curve, and the TPR and PPV (positive predictive value, or precision) vectors for the precision-recall curve. Also evaluate the 11-point interpolated precision, the break-even point, averaged precision, area under the ROC curve, and its standard error (see Section 2.3).

- V4: Format the results using Cheetah templating (Cheetah 2008) to produce the HTML of the output file. Graph the ROC curve and precision-recall curve, and the performance statistics. Also draw the distribution of document scores for relevant and irrelevant documents. To analyse the feature space, apply the feature selection criterion to the whole initial corpus, and report a table of feature weights, the aggressiveness of selection (ratio of features that would be selected to features that occur at least once in the initial corpus), and the histogram of the weights of the would-be-selected features. The reported feature statistics differ slightly from those within each cross-validation fold, where only 90% of the initial corpus is being used.

To calculate averaged precision, the program loops through the members of the initial corpus by decreasing classifier score, and adds the precision at each point where a relevant PubMed ID is encountered, finally dividing by the number of relevant documents (Manning et al. 2008; Chapter 8).

To calculate the distributions of document scores in the relevant and irrelevant components of the initial corpus, the program uses Gaussian Kernel Density Estimation over 512 equally-spaced points covering the range of article scores.

To calculate the histogram of feature weights, the program uses between 10 and 150 bins, with the number of bins within that range being a function of the range of feature weights R, number of features being binned $|\mathcal{V}'|$ (reduced feature space) and the inter-quartile range of the feature weights IQR:

$$\text{bins} = \left\lfloor \frac{R}{2 \cdot IQR \cdot |\mathcal{V}'|^{-\frac{1}{3}}} \right\rfloor \tag{3.11}$$

3.4.2 Initial corpora for evaluation

We make use of five different initial corpora, which vary widely in the number of relevant training examples, the diversity of topics among the relevant documents, the prevalence of relevant documents in the data (class skew), and the completeness of the training data. This prevents overfitting to a particular topic when selecting the best variant (Section 5.2.4). The initial corpora have been filtered to retain only valid PubMed IDs, and also only PubMed IDs which have "MEDLINE" status in the `<status>` element of the Medline record XML. This excludes, for example, "in-progress" records which do not yet have MeSH terms assigned, and "PubMed-not-Medline" records which fall outside of the Medline subject scope. Records with MEDLINE status have MeSH terms assigned, and their contents have been checked by the NLM.

The first four initial corpora are composites of a set of known relevant documents and a collection of random Medline records for irrelevant documents. Only the IEDB corpus (Wang et al. 2007) is a "gold standard" initial corpus, which was created by manually classifying a subset of Medline. For Radiology, AIDSBio, PG07 and Control, the relevant documents are combined with the Medline100K corpus, which consists of 100,000 random Medline records having completion dates up to 21 January 2007. We remove from Medline100K any overlap with the relevant corpus when making the initial corpus. We re-use Medline100K instead of taking independent samples, to avoid introducing more random variation into the comparison of the difficulty of the relevant corpora in Section 4.2.1. Performing cross-validation from the Web front-end does, however, take an independent sample from Medline on each use to make up irrelevant documents for validation (Section 3.4.1).

The random seed for shuffling the initial corpus before splitting is also fixed across all experiments to ensure that the same train/test splits are being used each time to compare different classifiers on the same initial corpus, in accordance with Section 2.3.5. However, performing cross-validation over the Web front-end will use a different random seed each time.

The initial corpora for cross-validation are as follows:

- PG07: Relevant documents consist of 1,652 pharmacogenetics evidence articles downloaded from PharmGKB (Hewett et al. 2002) on 5 February 2007. The articles are diverse (the only commonality is providing evidence for a drug and gene interacting), and the constructed initial corpus has low prevalence (1.6%). The scenario is that of database curation, where curators would review the results of filtering Medline.

- Radiology: Relevant documents consist of 67 PubMed IDs from a collaborator's bibliography, being radiology articles focusing on imaging of the spleen. It is the narrowest topic, and has very few relevant examples. Users may submit even fewer relevant articles to a filtering task. The scenario is that of an individual researcher seeking to extend a personal bibliography.

- AIDSBio: Relevant documents consist of 10,727 PubMed IDs from the intersection of the PubMed AIDS (NLM 2007a) and Bioethics (NLM 2007c) subsets on 19 October 2006. It has a large number of training examples and relatively higher prevalence. Each subset is defined by a complex PubMed query with a large number of results, so we expect AIDSBio to be relatively complete, leaving few documents in Medline discussing both AIDS and bioethics to be included by chance in the irrelevant corpus.

- Control: "Relevant" documents consist of a random sample of 10,000 PubMed IDs with completion dates up to 21 January 2007. These should be indistinguishable from the Medline100K documents, providing baseline performance when testing the finished classifier.

- IEDB: A gold standard initial corpus of 5,680 relevant and 15,132 irrelevant PubMed IDs (20,812 in total), being the manually classified results of a complex PubMed query for Immune Epitope Database evidence articles (Wang et al. 2007). Because the irrelevant documents arise from the same PubMed query, they are more similar

to the relevant documents than a random sample of Medline, raising the difficulty of the classification problem. The IEDB corpus is a filtered version of the original corpus in Wang et al. (2007), which contained 5,712 relevant and 15,198 irrelevant articles (20,910 in total). Our IEDB corpus excludes 98 records which did not have MeSH terms assigned.

3.4.3 Evaluation of variants of the classifier

The classifier formulation may be varied in three characteristics: smoothing method, feature selection method, and feature space. We aim to select a variant that performs well on average over the corpora of the previous section, without substantially under-performing against other variants on any particular case. To rank variants overall we use the mean averaged precision (MAP), where cross-validation yields Averaged Precision on the AIDSBio, IEDB, PG07 and Radiology initial corpora, and the mean is taken across the four tasks. Assuming that the best-performing choice for each characteristic is independent of choices made for other characteristics, the best overall variant can be obtained simply by varying each characteristic in turn from some baseline classifier. Where we suspect dependence between two characteristics we vary both, for example by comparing smoothing methods with and without feature selection.

Feature spaces and feature selection affect what document features are used and smoothing affects how the feature weights are calculated. We therefore not only compare performance, but examine differences on the feature level using the histogram of feature weights and aggressiveness of selection, calculated as described in Section 3.4.1. The reported feature-level statistics are from the whole initial corpus, not one of the cross-validation folds.

Smoothing method variants

The three smoothing methods for estimating the model parameters used to calculate feature weights are Laplace, Background and split-Laplace (Section 3.1.2). This experiment, carried out in Section 4.1.1, compares them using averaged precision, but also includes ROC area statistics. Evaluation uses the WMQIA feature space (consisting of title/abstract words, MeSH terms and qualifiers, ISSNs and authors) and two feature selection condi-

tions: no feature selection — which may be expressed as the document frequency condition $N_i \geq 0$ — and gain ratio selection using $GR(C, F_i) \geq 2 \times 10^{-5}$. Smoothing methods may perform better with or without feature selection, because they differ most for features with very low frequency in relevant or irrelevant articles, also the features most likely to be excluded by feature selection.

Feature selection variants

This experiment, carried out in Section 4.1.2 compares feature selection thresholds on averaged precision and aggressiveness. The classifier uses the WMQIA feature space and the chosen smoothing method from the previous section (split-Laplace). For document frequency, we test $N_i \geq x$ for x equal to 0, 1, 2, 3, 4 and 8. For gain ratio, we test $GR(C, F_i) \geq y$ for threshold y equal to 10^{-6}, 10^{-5}, 2×10^{-5}, 10^{-4} and 10^{-3}. The IEDB classifier used a criterion of $IG(C, F_i) \geq 2 \times 10^{-5}$, equivalent to $GR(C, F_i) \geq 2.36 \times 10^{-5}$ because the initial corpus was nearly balanced. We also evaluate the combination of $DF \geq 2$ and $GR(C, F_i) \geq 2 \times 10^{-5}$, and the recommendation in Mladenic and Grobelnik (1999) of only selecting features that occur in relevant documents using the criterion $N_{ri} \geq 1$.

Feature space variants

The WMQIA feature space was used to compare smoothing methods and feature selection methods because it contains the most information. This experiment, carried out in Section 4.1.3, compares the predictive value of the components making up the WMQIA feature space: the Word, MeSH, Qual, ISSN and Author feature spaces. We also include the MQI (MeSH, Qual, ISSN) feature space used in an older version of the classifier (Poulter et al. 2008) to quantify the performance effect of the new features. We evaluate averaged precision under cross-validation on each of the four initial corpora, both without performing feature selection and with $GR(C, F_i) \geq 2 \times 10^{-5}$ feature selection, to determine whether selection is undermining the predictive value of certain feature spaces. We also list the aggressiveness of the feature selection. In addition, we consider two variations on the WMQIA feature space:

- WMQIA-filt — A filtered version WMQIA, in which individual occurrences of Word

features are discarded if they also occur within one of the MeSH terms of that record, reducing redundancy between the MeSH and Word feature spaces.

- WMQIA-simple — Instead of constructing and composing individual feature spaces, WMQIA-simple is obtained by concatenating record fields for title, journal, abstract, author and MeSH terms into a single string and splitting on spaces. No other processing is done.

Finally, we compare variations on the word-splitting algorithm described in Section 3.2.1, which indexes the title and abstract fields in order to construct the Word feature space. For this experiment, the classifier uses the Word feature space, split-Laplace smoothing of Bernoulli parameters, and no feature selection. We consider the following variations:

- Case-folding: Convert all word features to lower case. The usual tokeniser capitalises the first letter, but otherwise leaves case unchanged.
- Numbers: Include features that look like numbers. The usual tokeniser discards number-like features.
- Strip punctuation: After splitting, remove all non-alphanumeric characters. The usual tokeniser retains most punctuation, except at the beginning and end of words.
- Simple tokenisation: Take the title/abstract text, and split on spaces with no other processing.

3.4.4 Evaluation of the final classifier

Having selected the final classifier variant, we seek in Section 4.2.1 to identify the reasons for differences in performance between corpora. We perform cross-validation on the Radiology, AIDSBio, PG07, Control and IEDB corpora, and analyse differences with the aid of distributions of document scores and feature weights, the receiver operating characteristic, and the precision-recall curve.

To place the classifier in context, we also compare its performance to two other classifiers in Section 4.2.2. First, for all initial corpora, we compare the final classifier to the older

version of the classifier presented in Poulter et al. (2008), which may be characterised as a variant using an MQI feature space, Background smoothing, and no feature selection. This serves to quantify performance improvements in the new classifier. Secondly, on the IEDB initial corpus, we compare the final classifier to the performance statistic reported on the IEDB corpus in Wang et al. (2007), in order to compare the final classifier to an independent classifier on its own data.

The last of the performance tests, in Section 4.3, is a relative retrieval test, designed to evaluate the classifier on data which constitutes a representative sample of Medline. The test performance measures how well the classifier retrieves the relevant documents of the IEDB corpus, relative to the PubMed query that produced the IEDB corpus. The test corpus consists of Medline restricted to records with completion dates in 2004, which is the only year that falls within the date limits of all components of the IEDB's PubMed query. Training data for the classifier consists of pre-2004 relevant documents of the IEDB corpus, with irrelevant documents consisting of all 2004 Medline records. "All of 2004 Medline" as an unlabelled background in training corresponds to the rest-of-Medline approximation used in operation (Section 3.3.1). The trained classifier ranks all 2004 Medline records, and precision and recall are calculated as a function of rank, where the 2004 relevant component of the IEDB corpus are relevant test documents. To compare to a PubMed query that retrieved those relevant documents, we give it constant precision equal to prevalence of relevant documents in the 2004 section in the IEDB corpus, with recall following a straight line from 0% at the beginning, to 100% at the rank equal to the size of the 2004 section of the IEDB corpus.

In the test above, relevant test documents are present in the "all of 2004 Medline" provided as irrelevant training, but their true labels are concealed from the training step, preventing the classifier from overfitting to relevant test documents. In fact, using all-of-Medline under-fits to the data, because the presence of relevant documents in irrelevant training data can weaken constitutive feature-class associations, in proportion to their prevalence.

3.4.5 Evaluation of classifier efficiency

The final experiment found in Section 4.5 examines the efficiency of the classifier and compares it to timing statistics reported for other classifiers. We measure the time from

submitting a filtering task on the Web front-end to returning results. The time spent actually classifying Medline (as opposed to training the classifier or formatting results) is then calculated from the log timestamps, and used to work out the speed of the classifier in documents per second. We also look at the compression resulting from the variable byte encoding of feature vectors (Section 3.2.2).

Chapter 4

Results

4.1 Evaluation of classifier variants

This section presents the results of the experiments described in Section 3.4.3. The experiments use cross validation (Section 3.4.1) on the initial corpora of Section 3.4.2 to compare options for the parameter smoothing method, feature selection method and threshold, and feature space used to index Medline records. The purpose of the experiments is to identify the best overall variant to use as the final classifier, and to understand what causes the differences in performance between the options.

4.1.1 Smoothing methods

First, we choose between the Laplace, Background and split-Laplace methods described in Section 3.1.2, which smooth the estimates of the naïve Bayes Bernoulli parameters, influencing the weight w_i assigned to each feature in training. For each feature, Laplace smoothing adds for each class one fictitious document or pseudocount with and without the feature, while Background smoothing adds for each class a small pseudocount equal to fraction of all documents in which the term is present, and split-Laplace smoothing adds in total two documents with the feature and two without, dividing them between the two classes by prevalence. We use the experiment of Section 3.4.3 to compare the methods

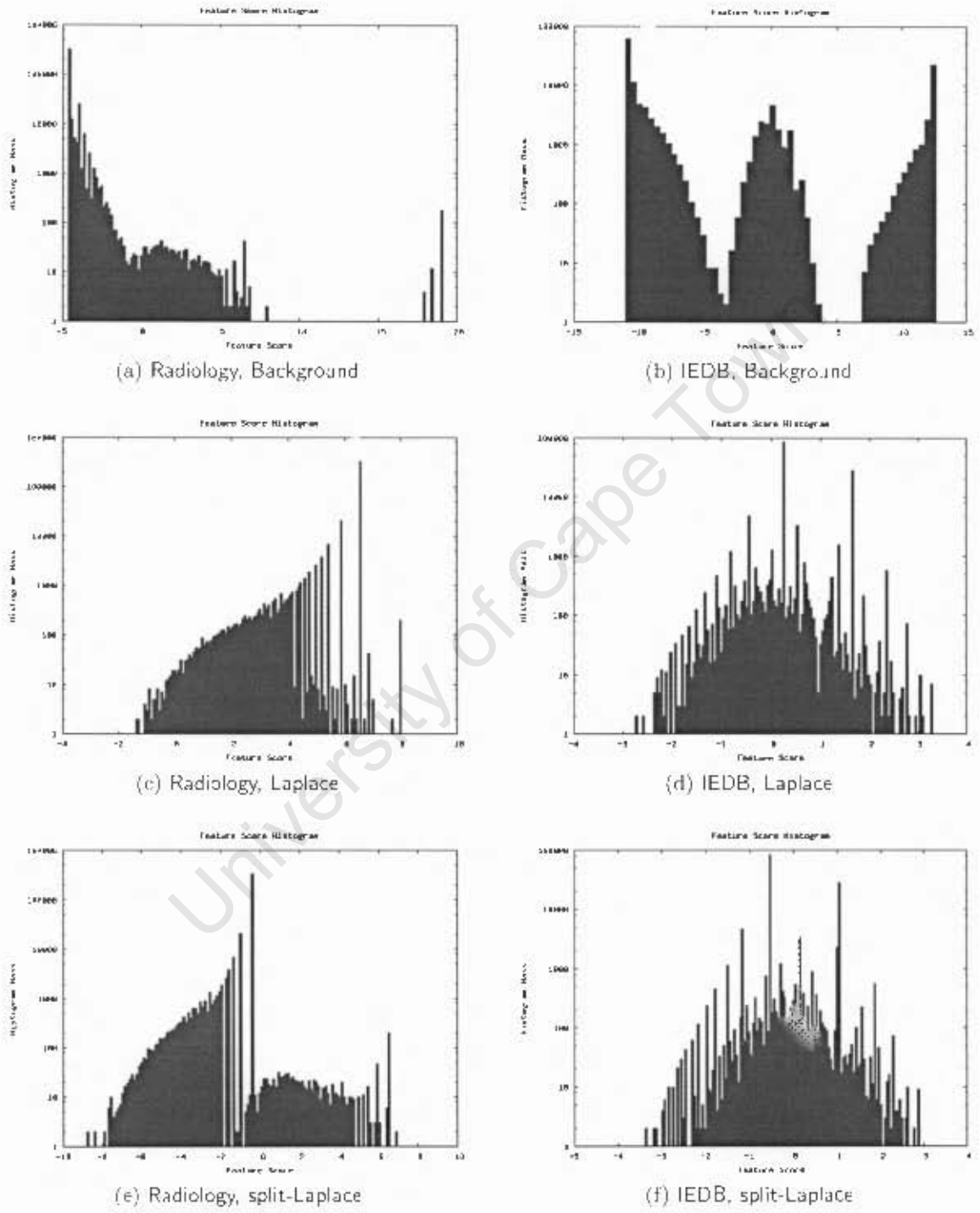
	AIDSBio		IEDB		PG07		Radiology		
Prevalence	0.097		0.273		0.016		0.001		
	AP	ROC	AP	ROC	AP	ROC	AP	ROC	MAP
Background	.911	.988	**.645**	**.814**	.740	.980	**.620**	.990	.729
Laplace	**.881**	.988	.694	.855	**.578**	.970	**.002**	**.721**	**.539**
split-Laplace	.914	.989	.708	.856	.752	.988	.684	.984	.765
Maximum	.914	.989	.708	.856	.752	.988	.684	.990	.765

(a) Without feature selection

	AIDSBio		IEDB		PG07		Radiology		
Aggressiveness	8.1		1.1		9.5		16.3		
	AP	ROC	AP	ROC	AP	ROC	AP	ROC	MAP
Background	.912	.989	**.644**	**.812**	.759	.984	.725	.992	.760
Laplace	.907	.989	.707	.858	**.719**	.981	**.031**	**.862**	**.591**
split-Laplace	.913	.989	.708	.856	.748	.987	**.682**	.985	.763
Maximum	.913	.989	.708	.858	.759	.987	.725	.992	.763

(b) With feature selection $GR(C, F_i) \geq 2 \times 10^{-5}$

Table 4.1: Performance of the Background, Laplace and split-Laplace methods of parameter smoothing. Averaged precision (AP) and the area under ROC curve use document scores collected from the 10 cross validation test folds. Mean of averaged precision (MAP) is taken over AIDSBio, IEDB, PG07 and Radiology. 4.1a uses no feature selection, while 4.1b uses the $GR(C, F_i) \geq 2 \times 10^{-5}$ criterion to select features in each validation fold. Reported aggressiveness is the ratio of features occurring at least once in the initial corpus to what would be selected by the criterion. The maximum of each column is listed in the bottom row, and bold font marks where performance was 0.02 or more below the maximum.



(a) Radiology, Background

(b) IEDB, Background

(c) Radiology, Laplace

(d) IEDB, Laplace

(e) Radiology, split-Laplace

(f) IEDB, split-Laplace

Figure 4.1: Histograms of feature weights (w_i) for different smoothing methods, comparing Background, Laplace and split-Laplace smoothing on the Radiology corpus (left) and IEDB corpus (right). These features from the WMQIA feature space occur at least once in the initial corpus, with weights calculated by Equation 3.5. Tall spikes are composed of low-frequency features, for which many features have exactly the same occurrence counts.

under cross validation. The averaged precision may be interpreted as the expectation of precision at the rank of a randomly selected relevant document. ROC area may be interpreted as the average recall (true positive rate) over all values of false positive rate, or as the probability of ranking a random pair of documents (one relevant, one irrelevant) correctly. Using the mean averaged precision in Table 4.1 Laplace smoothing performed the worst on average (because of the class skew in most corpora), then Background smoothing, and split-Laplace smoothing performed the best — both with and without feature selection. Split-Laplace smoothing will be used in the comparisons of feature selection methods and feature spaces.

To understand why the performance of the smoothing methods differ, we identify the cases where a particular method under-performs relative to the others. To this end, Table 4.1 marks in bold instances where the averaged precision or ROC area statistics are 0.02 or more below the maximum for the column. Examining Background smoothing first, on the IEDB corpus it results in averaged precision 0.05 to 0.07 lower than that for Laplace smoothing, and poor performance on the IEDB corpus using Background smoothing was also noted in Poulter et al. (2008). The main difference between Background smoothing and Laplace or split-Laplace is that its pseudocounts a_i are much smaller — around 10^{-5} compared to 1 for Laplace smoothing. The smaller pseudocount means weaker smoothing, producing more extreme weights for features found only in one class during training, which for the IEDB corpus occur around ± 10 in the histogram of feature weights in Figure 4.1b. The extreme scores are in contrast to the stronger smoothing of Laplace and split-Laplace in Figure 4.1d and Figure 4.1f, where the most extreme weights are around ± 3. On the Radiology corpus, Background smoothing had averaged precision 0.064 below split-Laplace smoothing without feature selection, but with feature selection it outperformed split-Laplace smoothing by 0.043. The reason for this is that feature selection eliminated most of the over 450,000 of the extreme negative-scoring features visible on the left side of the feature histogram for Radiology in Figure 4.1a, while leaving the positive-scoring features untouched. Eliminated features all had 0 occurrences in relevant articles and a few occurrences in irrelevant articles, making them uninformative for classification, but the small pseudocount in background smoothing had nonetheless given them extreme negative scores. We conclude that exaggeration of negative feature weights can cause poor performance with background smoothing in the absence of feature selection.

We now examine Laplace smoothing, which is commonly used in naïve Bayes classification.

However, classifiers are traditionally trained with balanced initial corpora like the IEDB whose prevalence of relevant documents is 0.27 (close to 0.5), rather than emulating the skew of using all of Medline for irrelevant training data, which we have done to different extents for AIDSBio, PG07 and Radiology. In Table 4.1 Laplace smoothing degrades relative to split-Laplace smoothing as class skew increases, starting out similar on the relatively balanced initial corpora of IEDB and AIDSBio, but performing substantially worse on PG07, and completely disrupting classification on Radiology. On Radiology, averaged precision is only 0.002, which is twice the 0.001 prevalence of relevant articles, making it about twice as precise as ranking the documents randomly. What happens is that, as explained in Section 3.1.2, Laplace smoothing substantially biases feature weights in favour of the low prevalence class, with the effect increasing the more skewed the classes. For Radiology this results in virtually all of the features being positive-weighted in the histogram of Figure 4.1c despite most of them not occurring in any relevant documents. With split-Laplace smoothing, the pseudocounts are adjusted for class skew to correctly give those features the negative weights visible in the histogram of Figure 4.1e. Introducing feature selection improves substantially the performance of Laplace smoothing in Table 4.1b, but even so it only reaches an averaged precision of 0.03 on Radiology. The feature selection eliminates mostly rare features occurring only in irrelevant examples, which are the features most affected by Laplace smoothing bias under high class skew. Medline often has even lower prevalence of relevant documents than the 0.001 for Radiology, rendering Laplace smoothing unsuitable.

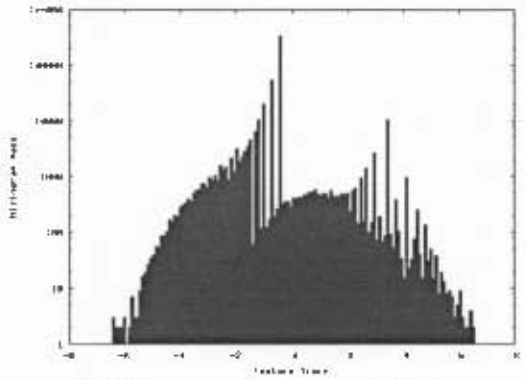
Finally we examine split-Laplace smoothing, which has the highest mean averaged precision in Table 4.1. It reduces to Laplace smoothing on balanced corpora like IEDB, where the weight histograms for the two methods in Figure 4.1d and 4.1f are nearly identical. Split-Laplace smoothing however outperforms Laplace for skewed classes, because splitting pseudocounts according to class ratio prevents features found only in irrelevant documents from obtaining large positive weights. Split-Laplace smoothing has higher averaged precision than Background smoothing except for the cases of PG07 and Radiology with feature selection, where it is lower by 0.011 and 0.043 respectively. There, the most of the exaggerated negative weights due to Background smoothing were eliminated by feature selection, and exaggerated positive weights may have pushed relevant articles further up the ranks to raise precision. However, exaggerating positive weights during filtering would raise the false positive rate, in the end causing lower precision.

	AIDSBio		IEDB		PG07		Radiology		
All features	497097		176882		494949		483999		
Features in relevant	36743		68735		33194		2057		
	AP	Agr	AP	Agr	AP	Agr	AP	Agr	MAP
$N_i \geq 0$	.914	-	.708	-	.752	-	.684	-	.765
$N_i \geq 1$	.914	1.0	.708	1.0	.752	1.0	.684	1.0	.765
$N_i \geq 2$	.914	3.3	.706	2.8	.748	3.3	.679	3.3	.762
$N_i \geq 3$	.914	5.3	.705	4.2	.745	5.3	.677	5.4	.760
$N_i \geq 4$	.914	6.9	.704	5.6	.739	6.9	.672	6.9	.757
$N_i \geq 8$	.914	11.1	.700	9.7	.729	11.3	.667	11.2	.753
$GR \geq 1 \times 10^{-6}$	.914	1.0	.708	1.0	.752	1.0	.682	1.0	.764
$GR \geq 1 \times 10^{-5}$	.914	5.9	.708	1.1	.750	7.3	.684	11.2	.764
$GR \geq 2 \times 10^{-5}$	.913	8.1	.708	1.1	.748	9.5	.682	16.3	.763
$GR \geq 1 \times 10^{-4}$	.914	27.6	.708	3.9	.742	16.3	.671	51.9	.759
$GR \geq 1 \times 10^{-3}$	.928	218.3	.692	257.8	.731	156.5	.576	231.4	.732
$\geq 2, \geq 2 \times 10^{-5}$	.913	9.9	.706	4.0	.744	11.8	.679	16.4	.761
$N_{ri} \geq 1$	.908	13.5	.706	2.6	.729	14.9	.491	235.3	.709
Maximum	.928		.708		.752		.684		.765

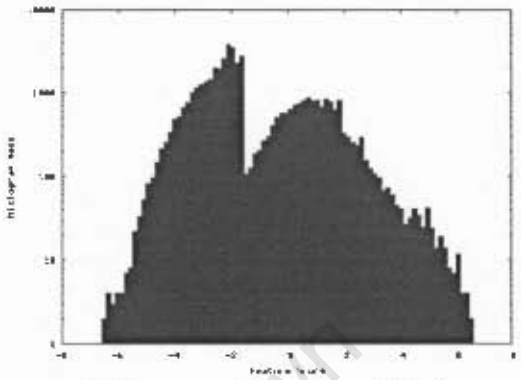
Table 4.2: Performance of feature selection criteria by Averaged Precision (AP), using the WMQIA feature space and the split-Laplace smoothing method. Document frequency of feature i is N_i and gain ratio is $GR(C, F_i)$, while $N_{ri} \geq 1$ selects all features that occur in at least one relevant article. Mean averaged precision (MAP) is taken over AIDSBio, IEDB, PG07 and Radiology. Reported aggressiveness ("Agr") is the ratio of features occurring at least once in the initial corpus to the number that would pass the selection criterion. The first row is the number of features with $N_i \geq 1$, and the second is the number occurring in at least one in relevant article ($N_{ri} \geq 1$). Prevalence is the fraction of relevant documents in the initial corpus, which would be the averaged precision obtained by ranking the documents randomly.

4.1.2 Feature selection

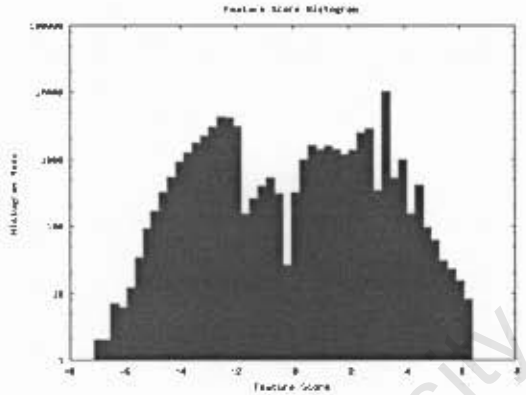
This section chooses a method and threshold for feature selection using the experiments from Section 3.4.3 to compare feature selection thresholds on averaged precision in Table 4.2. For all initial corpora except AIDSBio, the introduction of any feature selection degrades averaged precision over not performing selection (criterion $N_i \geq 0$). This result, for a naïve Bayes classifier with split-Laplace smoothing, stands in contrast to decision-tree classifiers where pruning of the feature space is necessary to avoiding the overfitting that leads to misclassifying test instances. Feature selection did, however, improve performance with the Background and Laplace smoothing methods in Table 4.1 by removing the features most vulnerable to biased parameter estimates. For AIDSBio, averaged precision was



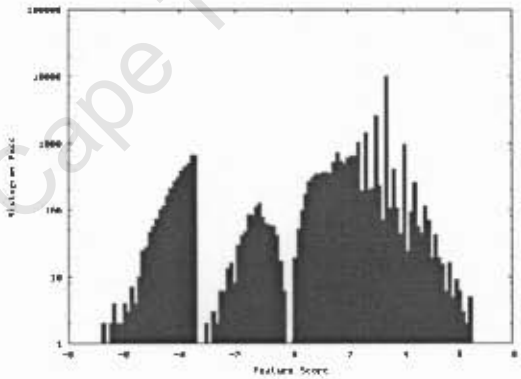
(a) Document Frequency: $N_i \geq 1$



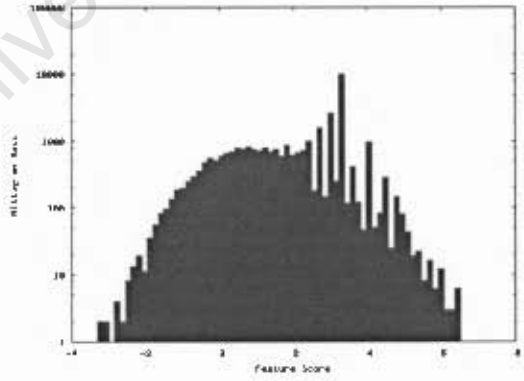
(b) Document Frequency: $N_i \geq 8$



(c) Gain Ratio: $GR \geq 2 \times 10^{-5}$



(d) Gain Ratio: $GR > 1 \times 10^{-4}$



(e) Features from relevant articles: $N_{ri} > 1$

Figure 4.2: Histograms of feature weights (w_i) using different feature selection thresholds. These are features found in the PG07 initial corpus, under the WMQIA feature space and split-Laplace smoothing of parameters. Sub-plots show which features are retained by different feature selection thresholds for document frequency or gain ratio.

roughly the same for all feature selection criteria, except for the stringent $GR(C, F_i) \geq 1 \times 10^{-3}$ criterion which increased averaged precision by 0.014 but substantially degraded it on the other corpora. Small feature spaces are desirable to reduce spurious feature-class associations (overfitting), but the split-Laplace smoothing seems to have produced good feature weight estimates in any case. To avoid loss of test performance, we choose a feature selection criterion that reduces the feature space as much as possible with minimal cost to performance.

We first choose between document frequency and gain ratio feature selection using Table 4.2. Document frequency generally degrades averaged precision to a greater degree than gain ratio for a given reduction in feature space, and conversely achieves less reduction in feature space for a given level of performance loss. For example, on the PG07 corpus the $GR(C, F_i) \geq 2 \times 10^{-5}$ criterion achieves a 9.5-fold reduction in feature space for 0.007 loss in precision, while $N_i \geq 3$ reaches that same loss with a 6.9-fold reduction. Usually, comparisons of feature selection methods would set exactly the same number of features, for example by using $N_i \geq 3$ and tuning the GR threshold to select the same number, before comparing performance. Earlier studies (Yang and Pedersen 1997) however back up the observation above that document frequency selection tends to be less effective in conserving performance, and so we will use just gain ratio selection in the final classifier.

To probe the differences between the feature selection criteria, we examine the histograms of feature weights for some of the criteria on the PG07 corpus in Figure 4.2. For document frequency selection, the histograms show that the $N_i \geq 8$ criterion in Figure 4.2b removes spikes visible in Figure 4.2a for $N_i \geq 1$. The spikes therefore correspond to clusters of low-frequency features with identical positive and negative occurrence counts. The $GR(C, F_i) \geq 2 \times 10^{-5}$ criterion in Figure 4.2c removes many of the same spikes, and $GR(C, F_i) \geq 1 \times 10^{-4}$ in Figure 4.2d additionally shows many features being removed with weights around -3. The lists of selected features showed that for $GR(C, F_i) \geq 2 \times 10^{-5}$ all features occurring only in relevant documents were selected, and that features occurring only in irrelevant documents needed at least 11 occurrences to be selected. With $GR(C, F_i) \geq 1 \times 10^{-4}$ irrelevant-only features needed at least 52 occurrences to be selected. The $N_{ri} \geq 1$ criterion in Table 4.2 selects only features found in relevant documents, and is based on the recommendation of Mladenic and Grobelnik (1999). The histogram of selected features for that criterion in Figure 4.2e shows that the criterion almost exclusively selects features that indicate relevance, with little discrimination for irrelevant documents. Even

so, the $N_{ri} \geq 1$ criterion only substantially reduced averaged precision in Table 4.2 for the PG07 and Radiology corpora.

The IEDB classifier (Wang et al. 2007) also used feature selection, with a combination of $N_i \geq 4$ and $IG(C, F_i) \geq 2 \times 10^{-5}$, which is equivalent to $GR(C, F_i) \geq 2.36 \times 10^{-5}$, together with Laplace smoothing. The criterion was reported to reduce the feature space from 181,299 features to 20,509 (aggressiveness 8.8), while raising ROC AUC for cross validation on the IEDB corpus from 0.846 to 0.848. Averaged precision was not given. In relation to our results, the column for the IEDB corpus in Table 4.2 shows that the nearest criterion of $GR(C, F_i) \geq 2 \times 10^{-5}$ yielded only a 1.1-fold reduction in feature space. Most of the reported reduction would therefore have been due to document frequency selection. Table 4.2 uses split-Laplace smoothing, and gain ratio selection did not affect performance on IEDB. However, for Laplace smoothing on the IEDB corpus in Table 4.1, we did observe an increase in ROC area from 0.855 to 0.858 when introducing $GR(C, F_i) \geq 2 \times 10^{-5}$ feature selection.

Taking this all into account, we select for the final classifier the gain ratio criterion $GR(C, F_i) \geq 2 \times 10^{-5}$, which yields a substantial reduction in feature space without substantially degrading averaged precision. The greatest decrease in precision was 0.004 on the PG07 corpus, for a 9.5-fold reduction in feature space. When performing Medline filtering, the criterion will produce up to 83-fold reductions in feature space (Figure 4.8) by removing a large number of low-frequency features occurring in Medline but not in any relevant examples.

4.1.3 Feature spaces

Having settled on methods of smoothing and feature selection, we use the experiments from Section 3.4.3 to compare the performance of the component feature spaces of WMQIA (the combined feature space with Word, MeSH, Qualifier, ISSN and Author features), some variants on WMQIA, and variations on the word splitting algorithm described in Section 3.2.1. In terms of performance, the final WMQIA feature space attains the greatest mean averaged precision (0.765) in Table 4.3. Comparing the AP-0 (averaged precision without selection) and AP-2 ($GR \geq 2 \times 10^{-5}$ selection) columns shows that feature selection degraded averaged precision by 0.002 or less for all feature spaces. An exception was the

	AIDSBio			IEDB			PG07			Radiology			MAP
Prevalence	0.097			0.273			0.016			0.001			
	AP-0	AP-2	Agr	AP-0	AP-2	Agr	AP-0	AP-2	Agr	AP-0	AP-2	Agr	
Word	.802	.798	6.6	.690	.690	1.1	.690	.682	7.5	.543	.531	11.5	.681
MeSH	.913	.913	1.7	.629	.628	1.2	.629	.625	2.0	.628	.632	2.4	.700
Qual	.609	.610	1.0	.459	.460	1.2	.221	.221	1.0	.032	.032	1.0	.330
ISSN	.705	.696	2.4	.365	.366	1.2	.448	.447	3.7	.091	.090	5.0	.402
Author	.485	.456	19.8	.535	.535	1.1	.481	.499	24.6	.040	.038	378.7	.385
MQI	*.922*	*.921*	1.9	.641	.641	1.2	.687	.682	2.3	.671	.673	2.8	.730
WMQIA	.914	.913	8.1	*.708*	*.708*	1.1	*.752*	*.748*	9.5	*.684*	*.682*	16.3	*.765*
WMQIA-filt	.915	.914	8.3	.704	.704	1.1	.749	.744	9.7	.631	.633	17.0	.750
WMQIA-simple	.915	.914	7.5	*.708*	*.708*	1.1	.736	.731	8.5	.565	.577	15.2	.731
Maximum	.922	.921		.708	.708		.752	.748		.684	.682		.765

Table 4.3: Performance of different feature spaces by averaged precision in cross validation. The Word, Mesh, Qual, ISSN, Author, MQI and WMQIA feature spaces are described in Section 3.2.1, and the experimental WMQIA-filt and WMQIA-simple feature spaces in Section 3.4.3. "AP-0" refers to averaged precision without feature selection, and "AP-2" refers to the averaged precision with $GR(C, F_i) \geq 2 \times 10^{-5}$ feature selection. "Agr" is the aggressiveness of the feature selection. Split-Laplace smoothing is used in all cases. Mean averaged precision (MAP) is calculated from the "AP-0" values. The maximum in each column is provided in the bottom row, and the entry achieving that maximum is in italics.

Author feature space on PG07, where the averaged precision improved from 0.481 to 0.499 with highly-aggressive (24.8) feature selection.

Comparing the value of the Word, MeSH, Qual, ISSN and Author components of WMQIA, Table 4.3 shows that the MeSH feature space had higher averaged precision than the Word feature space on AIDSBio and Radiology, but lower on IEDB and PG07. Using mean averaged precision, the next most valuable components were ISSN, Author and Qual. The relative value of the ISSN and Author spaces depend on the corpus: ISSN is more valuable in AIDSBio and PG07, while Author is more valuable in the IEDB corpus. MeSH Qualifiers have the least classification value, probably because Medline only uses a hundred or so different qualifiers. Author features also performed poorly, probably due to rarity of occurrence of all but the most prolific authors. Author features also have a degree of homonymy: the same initials and last name may refer to two different people, making the feature less informative about the class. The journal ISSNs are unambiguous, but contribute little to classification because there is only one feature per article, which in turn means that relatively few journal ISSNs (compared to words) occur often enough to contribute to classification. None of ISSN, Author and Qual performed well on Radiology, which has very few relevant examples from which to estimate term frequencies. Interest-

	AIDSBio	IEDB	PG07	Radiology	MAP
Word	.802	.690	.690	.543	.681
With numbers	.802	.691	**.679**	**.519**	.673
With case-folding	.802	.689	.689	.535	.679
Stripping punctuation	.802	**.684**	**.665**	**.525**	**.669**
Simple tokeniser	.798	.695	.693	**.439**	.656
Maximum	.802	.695	.693	.543	.681

Table 4.4: Performance of variations on the word-splitting algorithm described in Section 3.4.3, using Averaged Precision from cross validation. The classifier uses split-Laplace smoothing, no feature selection and the Word feature space. Mean of averaged precision (MAP)is taken across the four cases. The bottom row lists the maximum for each column, and cases where AP is at least 0.01 below the maximum are marked in bold font.

ingly, the MQI (which includes features from MeSH, Qual and ISSN) feature space used in Poulter et al. (2008) outperformed the larger WMQIA set of features on the AIDSBio corpus, despite being a subset of WMQIA. The good performance of MQI on AIDSBio is likely due to AIDSBio being defined by a large PubMed query (NLM 2007a, NLM 2007c) which placed many conditions on permissible MeSH terms, while the other corpora were developed without reference to MeSH terms.

In developing the WMQIA feature space, we experimented with variants in feature extraction before settling on the final WMQIA space. The WMQIA-filt variant reduces redundancy by ignoring text words already represented in the article's MeSH terms — but it in fact degrades precision in Table 4.3, because the redundant terms account to some extent for multiple occurrences of a concept, much like multinomial Bayes accounts for multiple occurrences of a term. The WMQIA-simple variant concatenates all the Medline record fields and splits on spaces with no further processing. It matches the precision of WMQIA on the AIDSBio and IEDB corpora, has 0.013 lower averaged precision on PG07, and only substantially under-performs on Radiology (0.565 versus 0.684). The result shows that for English, simple word-splitting performs almost as well as more careful methods in some cases. The simple indexing approach also loses the ready-made phrase structure of the Author and MeSH record fields, but may be compensating with better estimates of feature frequencies resulting from the record fields being combined at a word level.

Finally, we experimented with variants of the tokenising algorithm used to construct the Word feature space. In Table 4.4, the final Word space has the greatest mean averaged precision. The simple tokeniser which performs no processing besides splitting on spaces

in fact has slightly greater averaged precision on IEDB and PG07, but degrades averaged precision on Radiology — reflecting the results of WMQIA versus WMQIA-simple. Introducing full case-folding has little influence on performance except a slight reduction in averaged precision on Radiology, and including numerical tokens reduces averaged precision by 0.011 and 0.024 on PG07 and Radiology respectively. Stripping punctuation characters by treating them like spaces also reduced averaged precision on all but the AIDSBio corpus.

4.2 Evaluation of the final classifier

The investigations of the previous section provided optimised choices for variable characteristics of the classifier. The final classifier variant uses split-Laplace smoothing, $GR(C, F_i) \geq 2 \times 10^{-5}$ for feature selection, and the WMQIA feature space. In this section, we evaluate its performance under cross validation on the five initial corpora to understand why it performs as it does on each corpus, using the experiments described in Section 3.4.4. We also compare its performance to the old classifier variant used in Poulter et al. (2008) and the IEDB classifier described in Wang et al. (2007), to identify the causes of performance differences between the classifiers.

4.2.1 Comparison of different initial corpora

To analyse differences in performance between corpora, we first look briefly at the distributions of document scores in Figure 4.3, which are of the same form as Figure 2.2. By aggregating the scores from the 10 test folds of cross validation, we obtain a score for every document in the initial corpus. Plotting the score distributions of relevant and irrelevant documents separately provides a visual indicator of the ability of the classifier to distinguish them. The distributions of relevant and irrelevant articles overlap more for the IEDB corpus than for AIDSBio, PG07 or Radiology. In this case it is due to the irrelevant articles in the IEDB corpus having a higher degree of similarity to the relevant articles, because they are both results of the same PubMed query — in contrast to the other data sets where the irrelevant articles were a random sample from Medline. For the Control corpus, the curves overlap completely, indicating no ability to distinguish the random sample of 10,000 random Medline records constituting the "relevant" class from

	AIDSBio	IEDB	PG07	Radiology	Control
Number relevant	10726	5680	1656	67	10000
Number irrelevant	99998	15131	99998	99998	99998
Prevalence	.097	.273	.016	.001	.091

(a) Characteristics of initial corpora

	AIDSBio	IEDB	PG07	Radiology	Control
Aggressiveness of selection	7.5	1.2	8.8	14.2	7.5
Averaged Precision	.914	.708	.748	.681	.092
Area under ROC curve	.9892	.8556	.9874	.9842	.5020
Standard Error of AUC	.0005	.0029	.0009	.0081	.0030
Break-Even Point	.875	.657	.697	.672	.093

(b) Performance of final classifier

	AIDSBio	IEDB	PG07	Radiology	Control
Averaged Precision	.924	.578	.693	.711	.090
Area under ROC curve	.9913	.7848	.9754	.9923	.4975
Standard Error of AUC	.0004	.0036	.0020	.0047	.0030
Break-Even Point	.884	.591	.652	.642	.089

(c) Performance of old classifier (Poulter et al. 2008)

Table 4.5: Cross validation performance of the final classifier and the old classifier for five different initial corpora. Table 4.5a lists properties of the Section 3.4.2 initial corpora, including the number of relevant and irrelevant articles, and the prevalence of relevant articles in the initial corpus. Performance statistics include averaged precision, the area under ROC and its standard error for independent samples, and the break-even point. The final classifier uses $GR(C, F_i) \geq 2 \times 10^{-5}$ feature selection, whose aggressiveness we report for the whole initial corpus.

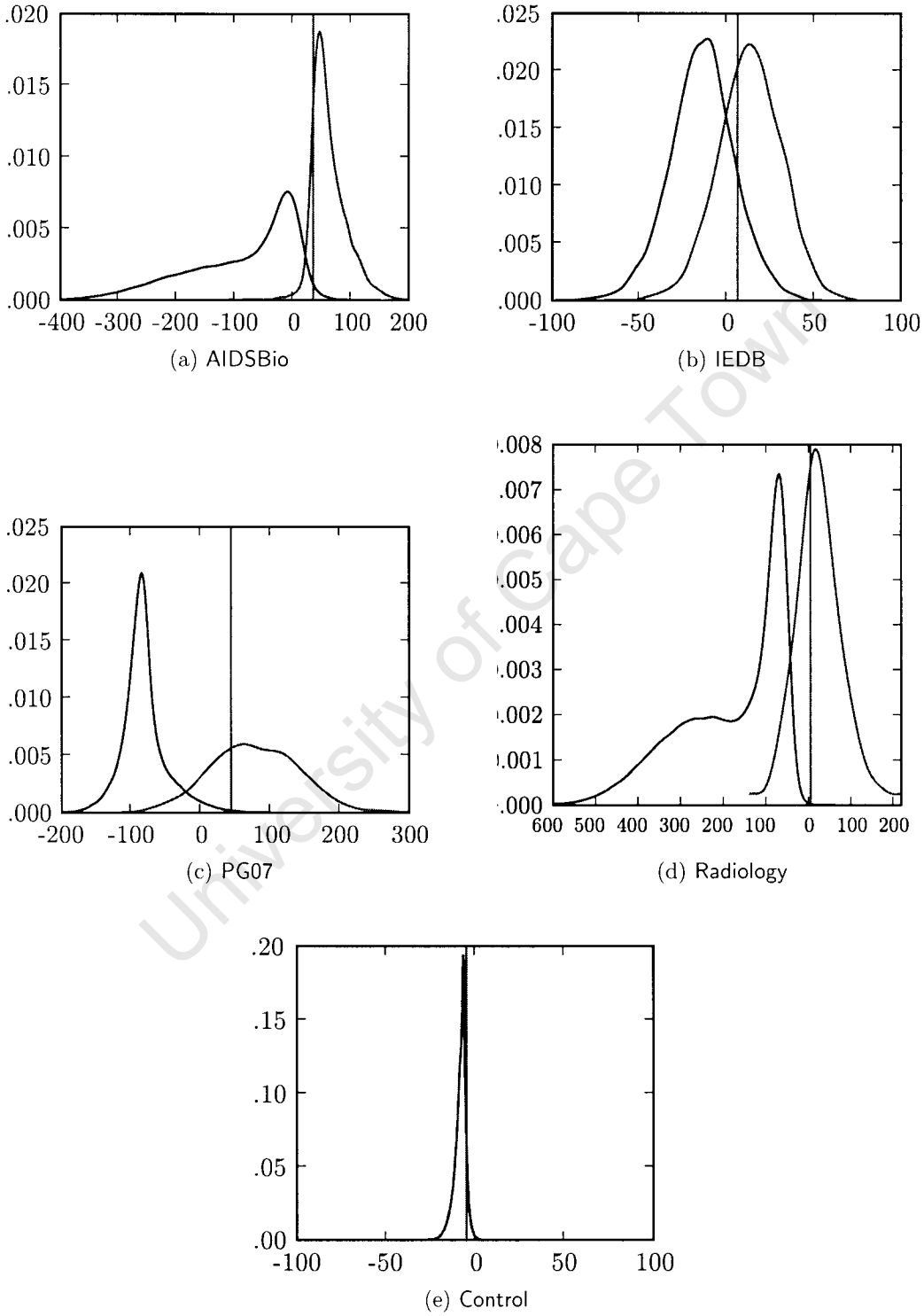


Figure 4.3: Distributions of relevant and irrelevant document scores, aggregated over the 10 test folds of cross validation, performed with the initial corpora AIDSBio, Radiology, PG07, IEDB and Control. Document score is on the horizontal axes and probability density on the vertical. The vertical bar marks the break-even point for the threshold where precision equals recall. The areas under curve to the left and right of break-even represent the true and false negative and positive rates at break-even. Some documents have scores beyond the displayed ranges.

the random records of Medline100K. In Poulter et al. (2008) the distribution curves for the Control corpus had multiple peaks, which corresponded to integer multiples of the weight of features which during training had occurred once by chance in the much-larger irrelevant part of the training data but not in the "relevant" part, and whose negative weights had been exaggerated by Background smoothing.

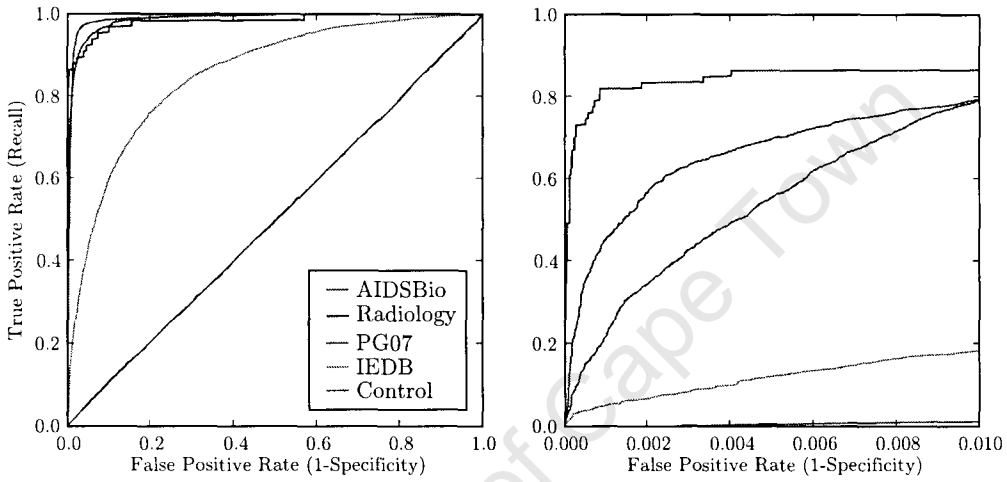


Figure 4.4: Receiver Operating Characteristic (ROC) curves from document scores in the five initial corpora, obtained using cross validation. The ROC graphs true positive rate (recall) against false positive rate (1-specificity), reflecting the trade-off between recall and specificity. Low false positive rates are of interest when filtering a large corpus like Medline, so we magnify the region below 1% false positives in the right-hand graph.

We now look at the Receiver Operating Characteristic (ROC) curves for each initial corpus in Figure 4.4, to identify which corpora are the most difficult and why. ROC curves, described in Section 2.3.4 are independent of class skew, so if two corpora have exactly overlapping ROC curves then they have equal difficulty, no matter that precision statistics may differ on account of class skew. As expected, the Control corpus attained worst-case performance, for which the true positive rate equals false positive rate at all thresholds, resulting in a straight-line ROC curve with 0.5 falling within the standard error of the area under curve (AUC) in Table 4.5b. This indicates no ability to distinguish between the "relevant" and "irrelevant" components of Control. In Table 4.5b, the AUC for PG07 of 0.9874 ± 0.0009 is significantly below those of Radiology (0.9842 ± 0.0081) and AIDSBio (0.9892 ± 0.0005), which do not differ significantly, indicating that PG07 relevant articles are harder to distinguish from Medline background. Radiology, however, has higher levels

of recall than PG07 and AIDSBio in the region of low false positives rate on the right of Figure 4.4, which is the region of the ROC curve most important for filtering Medline, indicating that Radiology is in fact the easiest classification problem. On the IEDB initial corpus, the AUC of the final classifier in Table 4.5b is 0.8556 ± 0.0029 — well below the AUC values for AIDSBio, PG07 and Radiology. Along with the article score distributions of Figure 4.3, this indicates that in the IEDB initial corpus the relevant articles are more difficult to distinguish from the irrelevant ones (independent of class skew), because both are results of the same PubMed query.

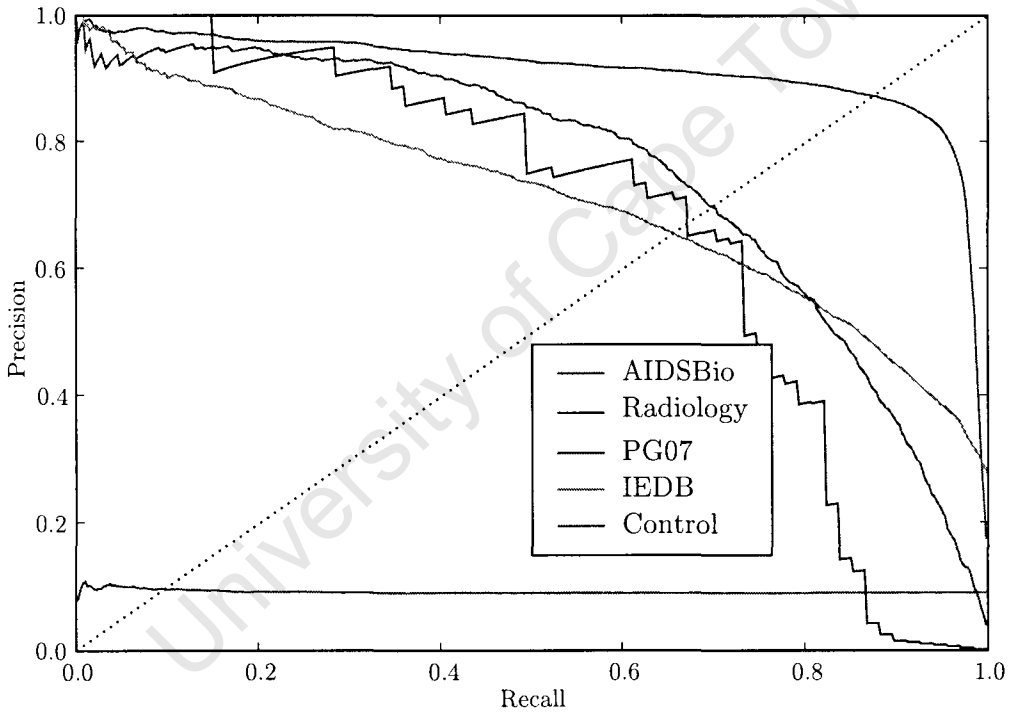


Figure 4.5: Precision-recall curves for five initial corpora, obtained by aggregating test document scores from the cross validation folds. Lower decision thresholds produce higher in higher recall and lower precision. The break-even points marked in Figure 4.3 correspond to where the precision-recall curve intersects the diagonal line.

Lastly, we analyse the difficulty of the initial corpora from the perspective of precision and recall. The curves of precision at different levels of recall are plotted in Figure 4.5, and are summarised by the averaged precision statistic from Table 4.5. As expected for the Control corpus, the precision at all levels of recall was roughly equal to the 9% prevalence of relevant

articles (Table 4.5a), meaning that top-ranked articles have roughly the same proportion of "relevant" documents as would a random sample from the corpus. Ranking the other corpora from easiest to hardest by averaged precision yields: AIDSBio (0.914), PG07 (0.748), IEDB (0.708) and Radiology (0.681). The ranking differs from what we observed in the ROC curves, where Radiology was determined to present the easiest classification problem and IEDB the hardest. The difference can be attributed to the influence of class skew (prevalence of relevant articles), on which precision has a direct dependence. The low prevalence of 0.001 in Radiology results in lower precision for a given false positive rate than the prevalence of 0.273 in IEDB.

4.2.2 Comparison to previous results

Using Table 4.5 we now compare the old classifier from Poulter et al. (2008) to the final classifier to understand where and why performance differs. The old classifier used Background smoothing, no feature selection, and the MQI feature space. In terms of both averaged precision and ROC area, the final classifier improves cross validation performance on IEDB and PG07, but degrades performance on AIDSBio and Radiology. In particular, the new classifier improves performance on the IEDB corpus over the old classifier, with averaged precision rising from 0.578 to 0.708, and ROC rising from 0.7848 ± 0.0036 to 0.8556 ± 0.0029. In the case of AIDSBio, the old classifier uses the MQI feature space, which outperforms WMQIA on AIDSBio in Table 4.3 because MeSH terms played an important role in the PubMed query defining the relevant AIDSBio documents. In the case of Radiology, the old classifier used Background smoothing, which outperforms split-Laplace smoothing on Radiology in Table 4.1 because the exaggeration of positive feature weights helped recall but did so without raising false positives during cross validation. These were exceptional conditions, and we expect the WMQIA feature space and split-Laplace smoothing in general to perform better than the MQI feature space or Background smoothing.

Lastly, we compare the final classifier on the IEDB corpus to the results reported for the IEDB classifier in Wang et al. (2007), where the evaluation was based on ROC curves. Our IEDB corpus excludes 98 documents in the original that had no MeSH terms. The IEDB classifier in Wang et al. (2007) used naïve Bayes (paper was ambiguous on whether the model was multivariate Bernoulli or multinomial), Laplace smoothing, and a combination of $N_i \geq 4$ and $IG(C, F_i) \geq 2 \times 10^{-5}$ for feature selection. The feature extraction algorithm

was similar to the WMQIA-simple variant, with little processing other than splitting on words (Wang 2007, *personal communication*). The final classifier in this work achieved an ROC AUC of 0.8556 ± 0.0029 on the IEDB corpus, greater than the ROC AUC of 0.848 reported in Wang et al. (2007). The results of Section 4.1 suggest that feature selection accounts for part of the difference, because we did not use $N_i \geq 4$ selection. Split-Laplace smoothing has little effect on the nearly-balanced IEDB corpus, and the WMQIA feature space performs just as well as WMQIA-simple on the IEDB data. The AUC of the IEDB classifier in Wang et al. (2007) did improve to 0.855, equalling the AUC of the final classifier reported here, when clusters of domain-specific features such as peptides, MHC alleles and position ranges were included in the feature space.

4.3 Performance in filtering Medline

This section tests the classifier under conditions similar to operation, to estimate its performance in Medline filtering. Comparing to the IEDB classifier showed that the classifier performs well in an ordinary database curation task on a Medline subset, but does not indicate how the classifier will perform when scaling up to classifying all of Medline. The AIDSBio, PG07 and Radiology initial corpora also are not representative samples of Medline, so the performance in cross validation does not predict performance on Medline at large. To test the classifier on a representative sample of Medline we use the relative retrieval test from Section 3.4.4. The test measures how well the classifier retrieves a known set of relevant documents from a one-year slice of Medline, relative to the knowledge-engineered PubMed query from whose results the known relevant documents were identified. Precision and recall for retrieving the known relevant documents under-estimate performance for retrieving all relevant documents, because the PubMed query has imperfect recall, leaving relevant documents which would be counted as irrelevant when retrieved by the classifier.

To perform the test, we prepared the following sets of Medline records. The test corpus is Medline restricted to the 783,028 records completed in 2004, chosen because it was within the date limits of all components of the PubMed query producing the IEDB corpus. For relevant documents, we start with IEDB corpus described in Section 3.4.2, which consists of 20,812 Medline records of which 5,680 were manually judged relevant and 15,131 irrelevant. 3,544 records have completion dates in 2004, of which 1,089 were judged relevant and 2,465

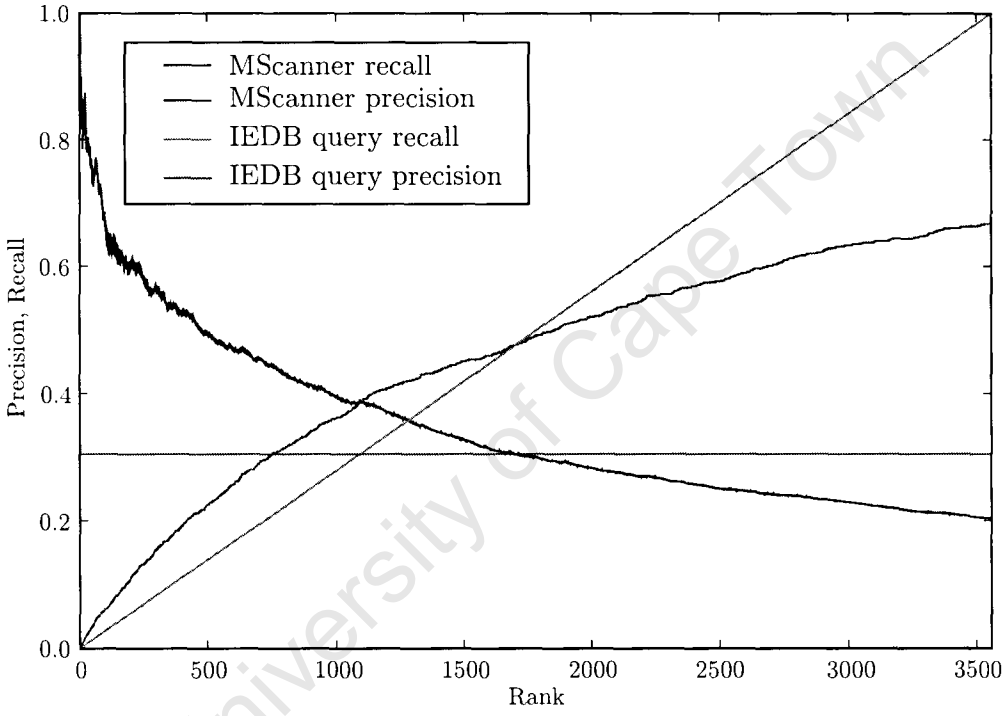


Figure 4.6: Relative precision and recall as a function of rank, on the task of retrieving the 2004 relevant documents of the IEDB initial corpus from 2004 Medline. This compares the final classifier to the PubMed query which generated the IEDB initial corpus. Constant precision is assumed for PubMed, because it does not relevance-rank results. The classifier is trained with pre-2004 relevant IEDB documents as relevant examples, and 2004 Medline as the approximately-irrelevant background to emulate the conditions of operation.

irrelevant, for prevalence of $1,089/3,544 = 0.307$. The 1,089 relevant IEDB documents are the known relevant test documents in 2004 Medline. To train the classifier, we use the 3,488 relevant IEDB record completed *before* 2004 for relevant training data, and "all of 2004 Medline" for irrelevant training data (equivalent to the rest-of-Medline approximation to irrelevant data used in whole-Medline filtering). The training step receives no information about which documents in 2004 Medline are relevant.

The trained classifier ranks 2004 Medline, and the ranks of the 1,089 known relevant documents produce the curves of precision and recall as a function of rank in Figure 4.6. The IEDB PubMed query which generated the IEDB corpus is assumed to have a constant precision of 0.307 at all ranks, and its recall follows a straight line from 0.0 to 1.0 at the point of 3,544 results (size of the PubMed result set). The classifier has greater precision and recall than the PubMed query up until about 1,700 results. When the classifier reaches 3,544 results, its recall and precision are 0.669 and 0.205. This means that the classifier, trained with about 61% of the relevant IEDB documents (3,488 out of the 5,680 relevant documents), retrieves about two-thirds of the relevant results that the IEDB query retrieved in 2004 for the same number of total results. The statistics of precision after returning N results are as follows: P10=0.818, P20=0.857, P50=0.745, P100=0.673, P200=0.602, P500=0.493. This means that amongst the top 500 ranked classifier results from 2004 Medline, 246 were members of the 1,089 known relevant documents in Medline. Of the other 254 documents, some would be unknown relevant documents, with the remainder being irrelevant documents.

4.4 Web front-end and retrieval service

Where Section 3.3.1 and Section 3.3.2 described the process of filtering Medline and the Web application architecture, this section examines the classifier from the user's perspective. We describe the user interaction with the Web front-end, and present results of filtering Medline via the Web front-end, using the relevant PG07 documents as input.

Classification tasks are submitted via the page shown in Figure 4.7. At a minimum, the user pastes a list of PubMed IDs into the box, enters a task name, and clicks "Submit Query". On the submission form, "Use all features" switches between the MQI and WMQIA feature space for improved precision at the cost of longer processing time; "Deletion code" is the

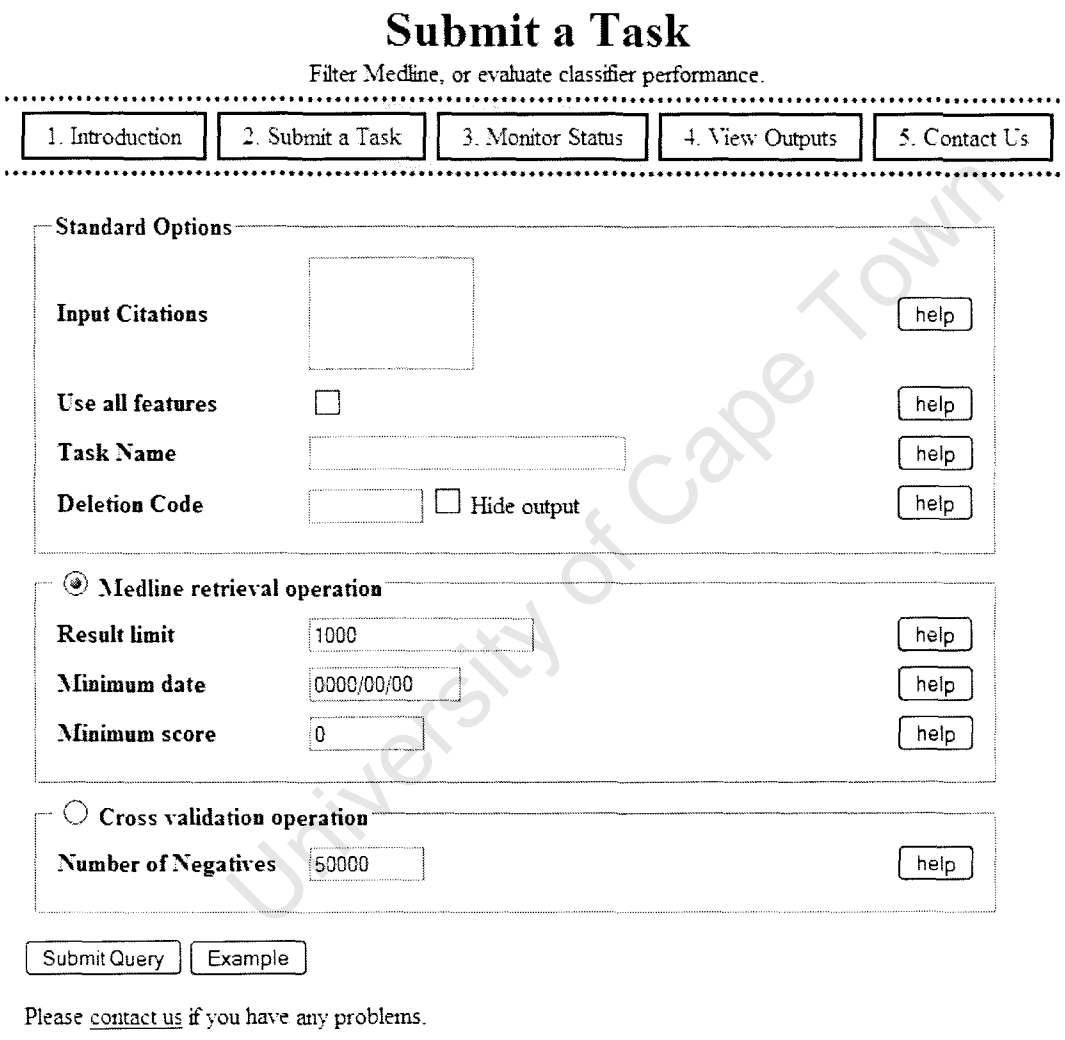


Figure 4.7: Form for submitting filtering/retrieval and cross validation tasks to the classifier. The form requires a list of PubMed IDs and a title for the task. The other parameters are optional.

Query Results

Number of results	1000	
Lowest scoring result	151.74648	
Abstracts of results	results.html	?
Abstracts of all results	ZIP file or all_results.html	?
Abstracts of input examples	inputs.html	?
PubMed IDs of results	results.txt	?
PubMed IDs of inputs	inputs.txt	?
Started at	2008/06/03 16:36:59 GMT	?
Finished at	2008/06/03 16:40:04 GMT	?
Feature score method	scores_laplace_split	?
Min Information Gain	2e-05	?
Base score	-69.3468709309	?
Prior score	-9.21215303107	?
Limit	1000	?
Threshold	0.0	?

Feature Statistics

Quantity	Relevant Docs	Irrelevant Docs
Number of documents	1656	16600062
Number of selected, occurring features	27717	44727
Total occurrences of selected features	179170	746258718
Selected features per Medline record	108.194	44.955
Of the considered feature types, 44728 features are selected out of 3703762 occurring at least once in training data. The aggressivity of selection is 82.806. The complete database lists 3703762 potential features.		

Figure 4.8: Cover page for the output of a Medline filtering operation for pharmacogenetics (black-and-white print version). Relevant documents from PG07 were provided as input. The page lists parameters of the classifier, statistics on the results, and the lower table provides statistics on the feature space.

Features with high TF.IDF

Features with TF.IDF above 0.2 or 0.3 could make good keywords. TF.IDF is term frequency times inverse document frequency, where we treat the set of input citations as a single document

TF-IDF	Type	Term	Term ID	Score	Pos	Neg
0.02	qual	genetics	799	3.66	1345	1656880
0.02	w	Polymorphism	11261	4.73	548	72062
0.02	mesh	Polymorphism, Genetic	11270	4.66	500	67796
0.01	w	Polymorphisms	5393	4.90	435	43909
0.01	mesh	Genotype	7862	4.41	488	84185
0.01	w	Gene	1420	3.04	803	713756
0.01	w	Genotype	5575	4.71	417	49910
0.01	w	Allele	6866	4.51	418	61351
0.01	w	Genetic	5269	3.10	493	312145
0.01	w	Associated	1562	2.04	621	1199820
0.01	w	Genotypes	20315	4.40	248	35815
0.01	mesh	Alleles	6887	3.95	263	60275
0.01	w	Variant	2683	3.80	271	72674
0.01	qual	metabolism	199	1.54	856	3087232
0.01	mesh	Polymorphism, Single Nucleotide	16633	4.92	205	17018
0.01	w	Human	442	1.76	524	1224339
0.01	issn	0960-314X	298455	7.47	138	854
0.01	w	Association	6427	2.44	351	379863
0.01	w	Variants	20226	3.71	238	67906
0.01	mesh	Gene Frequency	16630	4.16	203	36025

Figure 4.9: List of top features ranked by TF-IDF, found on the cover page. The high TF-IDF features are presented as an aid to identifying the best PubMed keywords to represent the training data.

Output Citations for pg07

Navigation: 1 2 3 4 Next

This report contains 250 citations. These are result citations.

Extra Features

Title contains	
Title/abstract contains	
Title/abstract does not contain	
Medline record date between	1900.01.01 and 2020.01.01
Abbreviated journal contains	
Author list contains	
Score is at least	-100
Order by	Score (decreasing)

Filter visible Show all citations Invert selection Help

Open visible in PubMed Open relevant in Pubmed Help

Table of citations (250 visible)

C	R	Score	PMID	Date	Au	Ab	Title	Journal
	1	402.12	12406645	2003.05.15	+	+	Genetic contribution to variable human CYP3A-mediated metabolism.	Adv Drug Deliv Rev
	2	300.29	16198658	2005.11.02	+	+	Cyclosporine markedly raises the plasma concentrations of repaglinide.	Clin Pharmacol Ther
	3	290.33	15536457	2004.12.02	+	+	A variant 2677A allele of the MDR1 gene affects fexofenadine disposition.	Clin Pharmacol Ther
	4	289.37	15116053	2004.05.27	+	+	Pharmacogenetics of acenocoumarol pharmacodynamics.	Clin Pharmacol Ther
	5	284.40	16509759	2006.04.26	+	+	Genetic polymorphisms of drug-metabolising enzymes and drug transporters in the chemotherapeutic treatment of cancer.	Clin Pharmacokinet

Figure 4.10: Topmost part of the first results page from filtering Medline for pharmacogenetics documents, using the PG07 relevant documents for input. There are 250 results per page. Clicking on the plus signs in the "Au" and "Ab" columns reveals the author list and abstract. The form at the top provides instantaneous sorting and filtering of the records within the page.

C	R	Score	PMID	Date	Au	Ab	Title	Journal
	1	402.12	12406645	2003.05.15	+	+	Genetic contribution to variable human CYP3A-mediated metabolism.	Adv Drug Deliv Rev
	2	300.29	16198658	2005.11.02	+	+	Cyclosporine markedly raises the plasma concentrations of repaglinide.	Clin Pharmacol Ther
	3	290.33	15536457	2004.12.02	+	+	A variant 2677A allele of the MDR1 gene affects fexofenadine disposition.	Clin Pharmacol Ther
	4	289.37	15116053	2004.05.27	+	+	Pharmacogenetics of acenocoumarol pharmacodynamics.	Clin Pharmacol Ther
	5	284.40	16509759	2006.04.26	+	+	Genetic polymorphisms of drug-metabolising enzymes and drug transporters in the chemotherapeutic treatment of cancer.	Clin Pharmacokinet
	6	279.89	12879168	2004.01.23	+	+	Polymorphisms of drug-metabolizing enzymes CYP2C9, CYP2C19, CYP2D6, CYP1A1, NAT2 and of P-glycoprotein in a Russian population.	Eur J Clin Pharmacol
	7	275.24	17496163	2007.09.20	+	+	Human hydroxysteroid sulfotransferase SULT2B1 pharmacogenomics: gene sequence variation and functional genomics.	J Pharmacol Exp Ther
	8	274.51	11503014	2001.09.20	+	+	Identification of functionally variant MDR1 alleles among European Americans and African Americans.	Clin Pharmacol Ther
	9	273.86	17047488	2007.02.06	+	+	Association of genetic polymorphism in ABCC2 with hepatic multidrug resistance-associated protein 2 expression and pravastatin pharmacokinetics.	Pharmacogenet Genomics
	10	272.80	9797796	1998.11.10	+	+	Bantu Tanzanians have a decreased capacity to metabolize omeprazole and mephenytoin in relation to their CYP2C19 genotype.	Clin Pharmacol Ther
	11	267.39	15557128	2005.05.12	+	+	Pharmacogenetic differences in response to albuterol between Puerto Ricans and Mexicans with asthma.	Am J Respir Crit Care Med
	12	265.66	7935325	1994.11.10	+	+	Genetic analysis of the Chinese cytochrome P4502D locus: characterization of variant CYP2D6 genes present in subjects with diminished capacity for debrisoquine hydroxylation.	Mol Pharmacol
	13	264.13	12419832	2002.12.20	+	+	A pharmacogenetic study to investigate the role of dietary carcinogens in the etiology of colorectal cancer.	Carcinogenesis
	14	262.63	12944498	2003.10.23	+	+	Novel functional polymorphisms in the UGT1A7 and UGT1A9 glucuronidating enzymes in Caucasian and African-American subjects and their impact on the metabolism of 7-ethyl-10-hydroxycamptothecin and flavopiridol anticancer drugs.	J Pharmacol Exp Ther
	15	262.20	16243813	2006.02.13	+	+	Effect of common CYP3A4 and CYP3A5 variants on the pharmacokinetics of the cytochrome P450 3A phenotyping probe midazolam in cancer patients.	Clin Cancer Res
	16	258.46	18056202	2008.05.07	+	+	Glutathione S-transferase T1 and M1: gene sequence variation and functional genomics.	Clin Cancer Res
	17	253.54	15572581	2005.08.25	+	+	Racial variability in haplotype frequencies of UGT1A1 and glucuronidation activity of a novel single nucleotide polymorphism 686C> T (P229L) found in an African-American.	Drug Metab Dispos
	18	247.82	16299241	2006.02.13	+	+	Association of CYP2C8, CYP3A4, CYP3A5, and ABCB1 polymorphisms with the pharmacokinetics of paclitaxel.	Clin Cancer Res
	19	246.61	17178267	2007.01.18	+	+	ABCB1 and cytochrome P450 genotypes and phenotypes: influence on methadone plasma levels and response to treatment.	Clin Pharmacol Ther
	20	244.84	17622941	2007.09.19	+	+	Effect of drug transporter genotypes on pravastatin disposition in European- and African-American participants.	Pharmacogenet Genomics
	21	244.48	15879416	2005.09.07	+	+	CD14 tobacco gene-environment interaction modifies asthma severity and immunoglobulin E levels in Latinos with asthma.	Am J Respir Crit Care Med
	22	241.77	17042920	2007.02.01	+	+	Effects of genetic polymorphisms of CYP3A4, CYP3A5 and MDR1 on cyclosporine pharmacokinetics after renal transplantation.	Clin Exp Pharmacol Physiol
	23	241.35	16890580	2006.09.08	+	+	Association of enzyme and transporter genotypes with the pharmacokinetics of imatinib.	Clin Pharmacol Ther
	24	240.29	11907488	2002.04.23	+	+	Low daily 10-mg and 20-mg doses of fluvoxamine inhibit the metabolism of both caffeine (cytochrome P4501A2) and omeprazole (cytochrome P4502C19).	Clin Pharmacol Ther
	25	239.64	11240981	2001.04.05	+	+	Frequency of single nucleotide polymorphisms in the P-glycoprotein drug transporter MDR1 gene in white subjects.	Clin Pharmacol Ther

Figure 4.11: Top 25 results from filtering Medline for pharmacogenetics, with the relevant PG07 documents used as input.

string that would have to be entered to remove the results from the outputs page; and "Hide output" will prevent the results from being listed on the outputs page. Configuration of the retrieval/filtering operation includes setting a limit on the number of results (defaulting to 1,000), constraining Medline to records completed after a certain date, and altering the score threshold from the default of zero. In certain circumstances only a few articles attain above-zero scores, so lowering the score threshold may be necessary. In the cross validation operation, a random sample of records is taken from Medline to serve as the irrelevant corpus, and "Number of negatives" modifies the size of the sample. The "Example" button fills in some Radiology PubMed IDs as an example use of the classifier. After submitting the task, a status page loads, which lists the tasks in the queue, how much time has elapsed since the task was started and a link to the future location of the results. It refreshes periodically, and displays a message when the results are ready.

Some of the fields in the submission page are there because the service does not require a login. If a login were used, the Web front-end would associate the query with the logged-in user, and only list results pages for queries performed by the logged-in user. The task name would not be strictly necessary, and neither would the deletion code and the "hide output" check-box. However, a login would require the user to invest time in creating an identity and saving a password before using the service, substantially raising the barrier to using the service. Having decided against requiring a login, the "task name", "deletion code" and "hide output" boxes made the interface more complex, but allow anonymous users to identify their tasks later and protect outputs against casual deletion or viewing by other users of the service if they so desire. Simply requiring a login would not result in greater privacy of results over the "hide output" feature, because the output files are static HTML and thus accessible by the correct URL. Protecting access to the results could be done by generating results pages dynamically, at the cost of a database, code complexity and longer response times. One might also have the static files stored elsewhere, with requests mediated by the Web front-end which would only read in allowed files, and pass the contents over to the Web browser.

We now examine the results of pasting the 1,656 relevant PubMed IDs from PG07 into the input box, selecting "use all features" and submitting the filtering task. The cover page in Figure 4.8 tabulates parameters and statistics on the filtering tasks, with "?" symbols that explain the corresponding entry when clicked. The cover page indicates that the task took 3 minutes and 5 seconds to complete (start to finish), and that the

lowest score was 151.95, so it is clear that there are results scoring above the 0.0 threshold that were excluded by the limit of 1000 results. In terms of parameters, it indicates that the classifier used split-Laplace smoothing, and $GR(C, F_i) \geq 2 \times 10^{-5}$ for feature selection. The base score of -69 is the score of an article with no features (all features failed to occur), and the prior score of -9.2 is the prior log odds ratio of encountering a relevant article. At the bottom of Figure 4.8, there are statistics about the documents and their features. The $1,660,002$ "Irrelevant Docs" for training consist of all of Medline except for the $1,656$ relevant examples. The cover page also has a feature statistics table, listing the number of distinct selected features in each class, the total number of features encountered and the per-document average. The aggressiveness of feature selection of 82.8 is also far greater than the 8.8 figure for PG07 in cross validation, due to the higher class skew requiring negative-weighted features to attain greater occurrence counts before being accepted based on gain ratio. For cross validation results, there is no cover page: just a single page listing comprehensive performance statistics, figures for ROC curve, precision-recall curve, document score distributions, and feature weight histograms as in Section 4.2.1.

Figure 4.9 is displayed at the bottom of the cover page for the results and shows features which are most suitable for keyword searching for the topic. The features are ranked by term frequency times inverse document frequency or TF.IDF (Joachims 1997). TF.IDF is designed to measure the importance of a term to a document, and is proportional to the number of times the term occurs in the document (term frequency) and inversely proportional to the number of documents in which the term occurs (document frequency). For our purpose, we use the number of occurrences of the feature in relevant documents as the term frequency — treating the relevant corpus as one big document — and use the number of occurrences in all documents as document frequency. High-TF.IDF terms have positive weights (w_i), but often the highest-weighted terms are quite rare amongst relevant documents, and thus have low TF.IDF. In the case of PG07, we observe that terms related to genetic polymorphisms have the highest TF.IDF.

Many of the high-TF.IDF terms in Figure 4.9 express similar concepts. If such synonymous terms were combined, for instance using stemming or combining identical MeSH and word features, it could make the parameter estimation more accurate for rare terms by increasing the occurrence counts. Retaining synonymous terms also exaggerates the probability of relevance when two or more synonymous terms are used in an article, because

the correlation between the terms is not accounted for. The exaggeration of probabilities is however a general issue with simple Bayes classifiers, as discussed in Section 2.2.4. On the other hand, an article using both terms is indeed more likely to be relevant than an article which only uses one, so keeping synonymous terms in a multivariate Bernoulli classifier can improve performance over combining them, because the combined term would only be counted once. This may not be the case in a multinomial document model, where multiple occurrences of a given term within a single document are accounted for.

Visiting the "results.html" file on the cover page leads to the results page, the top part of which is shown in Figure 4.10. Each page lists 250 results, although a zip file of a large HTML page with all 1,000 results may be downloaded instead. Each row of the results table shows the document score, a link to the PubMed record, the title of the article and the journal name. Clicking on the "+" symbol in the "Ab" or "Au" columns reveals the abstract or the author list under the row. The form at the top uses JavaScript to instantly filter and sort within the results table. Results may be ordered by score, title, journal or author, and the rows of the table filtered to show only entries containing specified text. The left-most "C" column is used for manual judging of the results. Clicking in each square marks the result as relevant or irrelevant, and the relevant ones can be viewed as a single result set in PubMed using the "Open Relevant in PubMed" button. If the results page is saved to disk and viewed locally, instead of being viewed on the remote server, there is also a function to save the manual classifications to a text file on disk. The "Help" buttons display explanations of the different functions.

Lastly, most of the highest ranking results are relevant. The top 25 results from the PG07 filtering task for pharmacogenetics evidence articles are shown in Figure 4.11. Of these, all but results 10, 16, and 25 have titles which indicate that the article presents evidence for a pharmacogenetic relationship, for an estimated P25 (precision at 25 results) of at least 0.88.

4.5 Efficiency of the classifier

This section examines the efficiency of the classifier, measuring its speed of Medline filtering using the measurements from Section 3.4.5. Medline consisted of 16,601,718 documents at the time. The inputs were Radiology, which has just 68 relevant training documents,

	Features	Radiology	PG07
Start-to-finish	WMQIA	186	185-204
	MQI	69	75
Classifying	WMQIA	168	163
	MQI	54	55
Articles-per-second	WMQIA	98819	101851
	MQI	307439	301849

Table 4.6: The start-to-finish time for filtering Medline, from submitting a task to returning results, and the time spent classifying the records, measured in seconds. This table compares the timing for Radiology and PG07. The articles-per-second statistic is calculated on time spent classifying, assuming 16,601,718 documents classified.

and PG07 which has 1,656. The classifier runs as a single-threaded process on a Sun Fire 280R server running Solaris 9 and having two 1200MHz 64-bit UltraSPARC-III processors with 8MB E-cache and 8192MB of RAM in 8 x 1024MB DIMMs. In Table 4.6, the Web front-end waits up to 204 seconds (3:24) to receive results when using the WMQIA feature space, and up to 75 seconds (1:15) with the MQI feature space. The lower figure around 185 seconds (3:05) for the response with PG07 occurs when certain files are cached by the operating system.

Part of the time from input to response is taken up with training the classifier and formatting of the results. In the Web front-end, the function to print a list of scores and statistics for selected features is disabled because it substantially increases the response time. The middle row of Table 4.6 lists times just for the core operation of evaluating scores for all Medline records and outputting a list of the highest scores. This requires takes roughly 168 seconds (2:48) to filter the 16,600,082 documents in the database using the WMQIA space, and 55 seconds using the MQI space. The corresponding speeds in the bottom row are roughly 100,000 and 300,000 documents per second respectively. As expected, the speed of classification is proportional to the number of features per record: Medline has an average of 43.8 WMQIA features per record, and 13.6 MQI features. It should be possible to increase the speed of the core operation by evaluating the scores in parallel on multiple nodes.

In the old version of the classifier presented in Poulter et al. (2008) it was reported that the classification responses took 60-90 seconds and classification took 30 seconds, using

the MQI feature space. When the new classifier uses the MQI feature space it takes 69-75 seconds overall, with 55 seconds spent classifying, making the core operation nearly twice as slow in the old classifier. This is due to variable byte encoding (Section 3.2.2) which compresses feature vectors. The new classifier uses 32-bit feature IDs to represent the WMQIA space, and compression shrinks the WMQIA feature stream from 3.67GB to 1.49GB, which halves the classification time when using the WMQIA feature space. If the new classifier is then set to use the MQI feature space from the old classifier, a 1.2GB stream of 32-bit MQI features is compressed down to 559MB, which is just slightly smaller than the uncompressed 600MB stream of 16-bit IDs in the old version. So with MQI features the new classifier using 32-bit feature IDs is faster than it would be without compression, but slower by 25 seconds than the old classifier using uncompressed 16-bit features.

The classifier described in Wang et al. (2007) classified "1,000 abstracts in less than 30 seconds", or 33.3 documents per second, making it roughly 3000 times slower than the final classifier described here. In Goetz and von der Lieth (2005), the PubFinder classifier was described as scanning "roughly half a million abstracts ... in 1-3 minutes" on a 22-node cluster, which is 2777-8333 documents per second or 126-379 documents per second per CPU, which is 264-794 times lower than the 100,000 documents per second of the classifier presented here. The speed difference is due partly to hardware differences and the diminishing returns of parallelisation, and partly to the speed-improving techniques discussed in Section 5.3. However, PubFinder service also appears unmaintained, as no results are being returned. In all, the final classifier here has cross-validation performance as good or better than the classifier developed especially for the IEDB, while being highly efficient and available as a service over the Internet.

Chapter 5

Discussion

This chapter discusses the significance of the results, and the complete work in relation to the goals set out in Chapter 1: to make a classifier for filtering all of Medline which is effective on a wide range of topics, available as a retrieval service, and highly efficient, while needing only relevant examples for input. We first discuss the overall effectiveness of the classifier and the resulting on-line service, and then explore the methodology behind the development of the classifier, and the reasons for particular design choices made regarding the classifier and experiments. Lastly, we look at the methods used to improve efficiency of the classifier, ways in which the classifier can be applied in operation, and possibilities for improving it.

5.1 Effectiveness of the classifier

In terms of performance, the effectiveness of the classifier is the precision and recall as a function of rank in the results of filtering for some topic. Measuring this requires either direct evaluation of the filtering results as in Rubin et al. (2005), or using test documents that are a representative sample of Medline. For subsets of Medline like the IEDB problem domain (Wang et al. 2007), cross validation can be done predictively, but prevalence is too low in Medline at large to construct an initial corpus. We used cross validation on initial corpora enriched for relevant documents to identify the best overall choices in parameter smoothing, feature selection and record indexing, and to analyse the difficulty of the dif-

ferent topics. The Web front-end to cross validation also allows the difficulty of new topics presented to the classifier to be measured. Comparing to the earlier version of the classifier in Section 4.2 we found that the new one improved performance overall, although the old methods worked better in particular circumstances (the MQI feature space on AIDS-Bio, and Background smoothing on Radiology). Comparing to the IEDB classifier on its subject domain, we found that the final classifier performs at least as well as the IEDB classifier on its own database curation task. The relative retrieval test in Section 4.3 then measured precision and recall for the IEDB topic using test data that were representative of Medline, and found that the classifier trained with a portion of the data retrieved about two thirds of the relevant documents identified by the knowledge-engineering query, for the same number of results. We also noted that precision was high, in at least the upper ranks when filtering Medline using PG07 relevant data in Section 4.4.

Performance of the final classifier, in terms of precision and recall, depends not only on the characteristics of the classifier, but on the coherence of the corpus of relevant documents used for training. If a single narrow topic is represented in the input, as is the case with Radiology where the topic is simply radiographic imaging of the spleen, the classifier easily identifies terms which distinguish it. The IEDB corpus however contains sub-topics, such as immune epitopes for emerging pathogens, category A-C pathogens, and allergens. The training step will then identify some terms as strongly predictive for all subtopics, but terms specific to one of the sub-topics will be less predictive because they do not occur in the others. As a result, relevant documents in each sub-topic would score lower than if the classifier had only trained for that sub-topic. In the limit of mixing a large number of sub-topics, the training data becomes indistinguishable from a random sample of Medline. As observed in cross validation with the Control corpus in Section 4.2.1, training with random documents retrieves more random documents. Regarding the relative retrieval test with IEDB documents in Section 4.3, more relevant documents might be retrieved by training and applying the classifier separately on each sub-topic.

Besides coherence of the topic, effectiveness also depends on the amount of training data provided, and the number of relevant documents left in Medline. In particular, narrow topics, for which few terms could indicate relevance (or topics with few sub-topics), need less training data is to obtain good performance. For example, the Radiology corpus about radiographic imaging of the spleen in cross validation has only 68 relevant documents for a prevalence of 0.001, yet its averaged precision is comparable (0.681 vs 0.748) to the broader

PG07 corpus with 1,656 examples and prevalence 0.016, where relevant documents provide evidence for pharmacogenetic relationships. Lastly, precision always drops when most relevant documents have been retrieved. Thus, if the training data already contains most relevant documents as may be the case with a particularly narrow topic, the precision will drop much more steeply than if many relevant documents remained in Medline to be retrieved.

A broader interpretation of the effectiveness of the classifier, subsuming performance statistics, is the total effort required to obtain a certain number of relevant results, which includes the effort of setting up the classifier and training data. For problems of identifying documents relevant to BIND, PharmGKB or the IEDB, the classifier here meets the objective presented in Chapter 1 of making supervised learning much easier to apply. The on-line service saves users the effort of constructing a custom classifier, filters all of Medline quickly so that it is not necessary to construct a Medline subset, and only requires relevant examples for input, which are more readily available than a manually classified corpus sampled from the domain of operation. The IEDB database did however get a Medline subset and training corpus "for free" from the pre-existing knowledge engineering filter for the database, which would not be available to databases that currently rely on user submissions or ad hoc search to identify relevant documents. Constructing a subset first and classifying it second does however have some advantages. Firstly, being smaller than all of Medline, it can be classified with slower non-linear classifiers. Secondly, a subset of Medline like the IEDB corpus is enriched for relevant documents, so the classifier's job is just to increase precision further, meaning that the IEDB classifier will obtain greater precision in the end than this classifier, simply because Medline at large has much lower prevalence of relevant documents. However, reviewing lower-precision results from classifying all of Medline with this service would require less effort than was involved in constructing a pre-filter query for a Medline subset, and manually labelling training data from the Medline subset.

Outside of database curation tasks, where supervised learning is the favoured solution, the effectiveness of the classifier at finding relevant documents can be compared to non-classifier alternatives. For example, individual researchers could upload relevant documents on some topic, using the classifier service to find more documents. This may be more or less effective than using retrieval services from Section 2.1, depending on circumstances. Collecting relevant PubMed IDs is more work than entering a search string, and the classifier takes 3 minutes to return results while search queries return instantly. On the other hand, if the

topic is broad enough not to be captured by a few search terms, collecting examples and letting the classifier decide what terms are important could be easier than using a large number of narrower queries, and could find articles that would otherwise be missed. An individual may however have only a few examples on the topic, which may not be enough to train an effective classifier. The classifier may even return zero results, when no Medline records have positive scores due to few predictive features. In such cases a similarity search like PubMed related articles (Lin and Wilbur 2007) may be more appropriate, finding for each relevant document the most similar other documents in Medline.

5.2 Experimental design

This section discusses and evaluate choices made in designing the classifier and experiments. Given the objectives in Chapter 1, and the classifier's performance in Chapter 4, we discuss the particular choice of classifier, and additional choices made in its design, such as the feature selection and extraction methods, and weigh up alternatives. We then examine the evaluation of the classifier.

5.2.1 Choice of basic classifier

Classifying all of Medline at high speed requires the classifier to be fast to apply, and fast to train. Linear classifiers such as SVMs, naïve Bayes and perceptrons can be made fast in application, as may decision trees. However, many classifiers from Section 2.2.3 have training steps involving iteration or optimisation, which would become markedly slower with more training examples, and so would not handle incorporating millions of rest-of-Medline documents as irrelevant training examples. We chose Naïve Bayes because it is fast to train, requiring time linear in the number of training examples, and makes it fast to use "rest of Medline" by pre-counting term frequencies in Medline. However, it is not ideal because not accounting for dependence between features leads to exaggerated document probabilities, which affects ranking. However, the requirement for a linear classifier that can accommodate millions of documents for training leaves naïve Bayes as the best choice.

The other choice we made is to use a multivariate Bernoulli document model, which only

accounts for presence or absence of each term. The alternatives would be a multinomial or multivariate Poisson model, which incorporate term frequencies (Section 2.2.3) and would be somewhat slower for having to keep track of them. However, most of the Medline record fields other than the abstract are suited to binary features: each MeSH term, journal ISSN, and author name is guaranteed to occur at most once in the record. Titles also tend to use each word once, and MeSH qualifiers may occur multiple times but contribute little to classification (Section 4.1.3). Only the abstract, although important, really has multiple occurrences of terms to account for. The multivariate Bernoulli model can to some extent account for multiple occurrences of a concept by retaining several synonymous terms, visible in the high-TF.IDF terms of Figure 4.9, for example "genetic polymorphisms" in MeSH, and singular and plural forms of "polymorphism" in the abstract. These occurrences would however be strongly correlated, increasing the exaggeration of probabilities referred to earlier.

5.2.2 Design choices around the classifier

On top of the basic classifier model, we made choices regarding parameter estimation, feature selection, and the indexing of Medline records. For parameter estimation, we proposed a novel modification to Laplace smoothing, which we called split-Laplace smoothing. This distributes the Laplace pseudocounts between the classes in proportion to prevalence, thus avoiding biasing feature weights toward the lower-prevalence class. Split-Laplace smoothing consistently outperformed Laplace smoothing in Section 4.1.1, as well as outperforming the earlier Background smoothing method we had presented in Poulter et al. (2008) which had used weaker smoothing to reduce bias at the cost of producing more extreme feature scores. While the classifier with Laplace smoothing, and most classifiers in general, improve with moderate feature selection, we found that the classifier with split-Laplace smoothing does not. Because split-Laplace smoothing is already the best performer, its bias correction is therefore effective enough that feature selection is no longer required for removing noisy features (such as the extreme weights resulting from Background smoothing) or biased features (resulting from Laplace smoothing), although it may still aid in reducing overfitting.

With feature selection, we used the gain ratio (Quinlan 1986) modification to the information gain metric to compensate for the low entropy of the class random variable under class

skew and thus make it easier to choose a cut-off usable across a range of conditions. We also found that selection to eliminate low document frequency terms reduces performance more than gain ratio selection for a given level of aggressiveness. We chose a gain ratio cut-off of 2×10^{-5}, which reduces the feature space without the substantial loss in performance observed using document frequency cut-off. It mostly excludes rare features found only in irrelevant training examples. We did not evaluate additional feature selection metrics found in Yang and Pedersen (1997) and Forman (2003), because their results showed that among the well-established feature selection metrics information gain (and therefore gain ratio) provided the best performance.

With the indexing of Medline records, we extracted features from abstract/title words, MeSH descriptors and MeSH Qualifiers to produce the WMQIA feature space. We found that using all of the component feature spaces resulted in the best overall performance. Individually, we found that MeSH descriptors and title/abstract words were the only fields to perform well on their own. Nonetheless, classifiers based on journal ISSNs, authors and MeSH qualifiers individually performed better than random ranking of documents, meaning that including them in the WMQIA space should aid ranking slightly over just using title/abstract and MeSH descriptors.

We did not use any methods to reduce the presence of synonymous features in the WMQIA feature space. The presence of two synonymous features predicting relevance in an article causes it to score more highly than if only one had occurred. Because the multivariate Bernoulli model only tracks presence/absence of features, it loses this multiple-occurrence information when features are combined. On the other hand, combining synonymous features can yield a more accurate estimate for the weight of the combined feature. The WMQIA-filt feature space modified WMQIA by, for each article, simply removing Word feature occurrences that were found within a MeSH for the article, thus removing multiple occurrences without improving frequency estimates, resulting in slightly worse performance. The WMQIA-simple feature space combined the Medline records fields for title/abstract, MeSH, author and journal name at the word level by concatenating them. Merging feature spaces at the word level reduced synonymy, but also improved frequency estimates for resulting word features. The result equalled the performance of WMQIA (with its synonymous features) on the IEDB initial corpus, but not on PG07 or Radiology. It may be that precision of MeSH phrase concepts (lost when split into words), and multiple occurrences of concepts (lost when features are clustered) becomes more important

on narrower topics. Word features can be further clustered by stemming them during indexing (Section 2.4.1), potentially improving estimates further by clustering similar words. However, stemming reduced performance in Wang et al. (2007).

5.2.3 Evaluation of the classifier

Using Medline or a random sample from Medline such as Medline100K for irrelevant documents in training produces lower performance compared to using a training set that is purely irrelevant. Incorrectly labelled training instances result in the classifier under-fitting, with some feature-class associations being weakened proportional to the prevalence of the unknown relevant documents. This under-fitting occurs when training using the rest-of-Medline approximation in operation, and in the relative retrieval test of Section 4.3 where all of 2004 Medline is used for irrelevant training data, including 1,089 relevant IEDB documents whose labels are unknown to the training step. The decision to use rest-of-Medline in training thus under-fits somewhat, but the benefit is classifying all of Medline while providing relevant examples, instead of classifying on a subset of Medline and having to manually label a training sample from that subset.

Using random documents for irrelevant data also influences test performance directly. In the test data, unknown relevant documents will rank higher than irrelevant documents but be labelled as false positives, reducing ROC area and averaged precision. Only 100,000 documents were sampled for Medline100K to limit their influence, while maintaining high-skew conditions for the purpose of designing the classifier. For example, if the prevalence of pharmacogenetics evidence articles was 0.0005 in Medline100K, this would produce 50 unknown relevant documents in addition to the known 1,656 known relevant documents in the PG07 corpus. However, in the relative retrieval test of Section 4.3, all of 2004 Medline (numbering 783,028 records) may have hundreds of relevant documents beyond the 1,089 relevant documents detected by the IEDB query, depending on the query's recall. Counting them as false positives in the ranking means the retrieval test over-estimates false positive rate and under-estimates precision.

The cross validation experiments produced point estimates of performance by micro-averaging across folds, without estimates of sample variation, and without significance testing between classifier variants. Instead, we compared the methods in Section 4.1 by

looking for substantial differences in performance between them on the same data, with sources of variation controlled by using the same train/test splits each time in validation. Paired permutation tests have been developed for comparing ROC area (Braun and Alonzo 2008) for two methods on the same sample, but not for the averaged precision we used for comparison. In information retrieval a pair of retrieval methods may be evaluated on hundreds of queries, and it is common to test for significant differences in mean averaged precision using a Wilcoxon signed-rank test that counts the number of times each method out-performs the other. However, the approach is not useful when there are just four binary classification problems (AIDSBio, IEDB, PG07 and Radiology). When comparing the difficulty of corpora for the final classifier in Section 4.2.1, we focused on comparing the relevant documents in Radiology, AIDSBio and PG07 by re-using the irrelevant background (Medline100K). In that case, the standard error reported for ROC area in Table 4.5 would be useful for detecting significant differences, as would the sample variation (Section 2.3.5) obtained by macro-averaging cross validation folds.

The results include one comparison of cross validation performance to an independent classifier, that of the IEDB (Wang et al. 2007), which used ROC area as the performance statistic. The IEDB classifier met the criteria of using Medline records, using a cross validation or train-and-test design, and making the test data available (either supplementary or on request). Ideally the test data would also be a representative sample from Medline, but all classifier evaluations on labelled test data have used a pre-filtered subset of Medline as their test domain. Aphinyanaphongs et al. (2005) and Rubin et al. (2005) performed cross validation on Medline records, but the original corpora were not available. OHSUMED (Hersh et al. 1994) is a standard Medline corpus. However, OHSUMED is designed for ad hoc information retrieval and marks a few relevant results for each of hundreds of terse queries, and so is unsuited to training and testing with many relevant documents as encountered in database curation. Of train-and-test evaluations of classifiers in the literature, test corpora were unavailable for Marcotte et al. (2001) and Donaldson et al. (2003), no evaluation was performed in Goetz and von der Lieth (2005), and the test corpus did not match the description in Suomela and Andrade (2005). The TREC Genomics (Hersh et al. 2005) categorisation task used train-and-test in a curation scenario for the Mouse Genome Informatics database. It used full-texts (as opposed to Medline records) and the corpus was enriched for relevant document (like that of the IEDB) with four distinct relevant topics inside, and so would not have been useful for predicting Medline filtering performance for one topic. The final classifier could in principle have been adapted to the TREC experi-

ment as a separate experiment, just using Medline record fields (not allowed in the TREC Genomics 2005 rules) instead of the full-text.

5.2.4 Overfitting

In Section 4.1 we make fixed, global choices about the classifier algorithm by comparing the performance of different variants. This is not the dynamic tuning discussed in Section 2.3.2 that requires a tuning hold-out set in each training fold of cross-validation. However, choosing the variant based on a single data set would pose a risk of constructing a classifier that performs poorly on new topics, effectively overfitting to the topic. For example, if we had used only the IEDB data set in Section 4.1.1, we would have chosen Laplace smoothing, even though it would perform poorly when given skewed training data in operation. The IEDB classifier (Wang et al. 2007) in Section 2.5 was optimised using just the IEDB data set, but the resulting classifier was only intended to be used on the IEDB topic and under balanced conditions, so overfitting was not a consideration.

In this work, we avoid overfitting to one data set by using four data sets of different topics for choosing the classifier (Section 3.4.2), because the classifier is intended to be used on any topic unlike the classifiers for particular databases. Choices that perform well on data sets with a wide range of characteristics will probably perform well in general. With enough corpora, it would even be possible to split the topics into "training topics" used to make the choices, and then demonstrate that the final classifier is still the best on "test topics" which were not used in the decision. This is to some extent the procedure followed after Poulter et al. (2008): we had chosen the old classifier using the AIDSBio, Radiology and PG07 corpora. However, after finding the IEDB data set, we discovered that MeSH features and background smoothing were not universally optimal. MeSH happens to outperform Word features for AIDSBio, and background smoothing works under class skew but falls down when the training data is balanced — insights which led to the final classifier. However, there may be corpora dissimilar to AIDSBio, IEDB, PG07 and Radiology, on which the final classifier will perform relatively badly, but could be improved with a modification to parameter smoothing or feature selection.

5.3 Efficiency of the implementation

Efficiency is vital when providing a service that classifies all of Medline, and was one of the main goals set out in Chapter 1. The timing results of Section 4.5 showed that the classifier takes about 3 minutes to return results using the WMQIA feature space and just over 1 minute with MQI features, making it the fastest implementation of a naïve Bayes classifier for Medline records in the literature, and comparable to Web services like NCBI BLAST. The final version of the classifier therefore meets the objective of being fast enough to use from a Web interface. To achieve efficiency we began with a simple Bayes classifier as discussed in Section 5.2.1, and proceeded to trade simplicity for greater speed. The resulting classifier has a simple conceptual underpinning, but performs complex pre-processing during Medline parsing to construct and maintain specialised data structures, to optimise execution time.

Techniques such as maintaining multiple representations of the data and compression of feature vectors in in Section 3.2.2 are well-known in the field of information retrieval. It was possible to adapt these techniques to a supervised learning context, thus speeding up the classifier. The methodology for choosing efficiency-improving techniques was to profile the classifier and identify the the step which was taking up most of the classification time. We would then identify a technique to make that step faster. The running time of the final classifier is dominated by a simple C program which reads feature vectors from the disk and sums them to produce document scores. The following techniques removed several bottlenecks present in earlier versions of the classifier:

- Using an article database means Medline records can be looked up to print the results, so the classifier is free to work with just PubMed IDs and feature vectors before that point.
- Using numerical feature IDs instead of the original character strings both makes feature vectors much smaller, and allows the training step to use fast vector operations for calculating the weight vector $\mathbf{w}$ and performing feature selection.
- Using a feature vector database means feature vectors are generated once during parsing and looked up as needed. The alternative of indexing text during classification is what would otherwise take days.

- Using a feature stream to list PubMed IDs and feature vectors is redundant to the database. However, it allows a small program in the C programming language to take only 55 seconds (MQI features) or 168 seconds (WMQIA features) to perform the crucial step of summing w_i feature weights for each of 16 million documents in Medline to produce document scores, and returning the top-ranking PubMed IDs. Using Python instead of C, this step took about 25 minutes for MQI features (predicting 75 minutes for WMQIA features).

- Tracking the number of occurrences of each feature in Medline during parsing immediately provides an N_i value for its term frequency in Medline. When using the rest-of-Medline for training, a simple vector subtraction gives the $N_{\bar{r}i} = N_i - N_{ri}$ counts for training. The alternative would be to count the features after the input was provided, which means processing Medline twice: first to count N_{ri}, and second to classify the records.

- Vacuuming the feature table (Section 3.2.3) reduces the WMQIA space from 12 million features to 3 million by removing features that occur just once or twice in all of Medline. These features don't affect performance, and the smaller table of features makes it four times faster to perform feature selection and feature weight calculation (taking about 5 seconds instead of 20).

The feature table provides two-way lookup between feature IDs and feature strings, and efficiency problems. Every feature encountered when parsing Medline is checked against the feature table: if present its count is incremented and if not a new feature ID is created with a count of 1. In earlier versions of the classifier (Poulter et al. 2008), there were only 25,000 MeSH terms and 17,000 ISSNs, which were maintained in an in-memory data structure that was saved to disk later. However, introducing Word and Author features caused the feature table to grow to over 12 million entries, taking up several gigabytes of memory. At that point, it made sense to store the feature table on-disk in a database. The database solved the memory problem, but introduced an efficiency problem. There was no issue in day-to-day operation because the classifier would only looks up the names of a few features that appear in the output, but when parsing Medline initially the on-disk database would take a predicted 19 days to perform the billions of insertions and increments of feature counts required, while the in-memory data structure needs only 2 days. In the final version, we have kept both the in-memory and on-disk data structures. The in-memory data structure is used once each year to pre-process the Baseline distribution

(NLM 2007b), and is then saved to a database on disk. This database is then used for day-to-day classification of Medline and addition of Medline updates. Redundant data structures are common in search engine technology: here we have adapted the technique for a large-scale supervised learning problem.

5.4 Applications of Medline filtering

Identifying relevant articles for the curators of biomedical databases is intended to be one of the primary applications of the classifier. Curators would extract relevant articles from their database, submit them to the on-line service, and review the results for more documents to add to the database. Ordinarily, a database would follow a procedure similar to that used by the IEDB (Wang et al. 2007) or ACP Journal Club (Aphinyanaphongs et al. 2005) where they would construct a filter to limit Medline to a subset, develop a gold standard training set, and develop a custom-built classifier on the Medline subset. Databases that already use supervised learning could also use the service to look for documents that were inadvertently excluded by the pre-filter that provides documents to their existing classifier. A classifier service which needs just relevant examples meets the goals of Chapter 1 for lowering the barrier to filtering Medline records using supervised learning.

Instead of constructing a PubMed query or other filter, the classifier service could also form a pre-filter on Medline, providing thousands of results to a stronger but more computationally expensive classifier or filter. For example, in Rubin et al. (2005) the results of filtering Medline were sent to a relationship-extraction method to filter for documents containing co-occurrences of drug names and gene names. In this case, the stronger filter is too slow to apply to all of Medline, and so requires a high-recall subset of Medline. Knowledge engineering could be performed to construct a high-recall filter, but training the classifier service from existing relevant examples would be easier in a database curation scenario. The results of Section 4.3 with the relative retrieval test suggest, however, that the filter queries can retrieve some relevant documents that would otherwise have low classifier scores.

The classifier service has potential for broader application beyond database curation. In general, anyone with documents on a particular topic can submit them to the classification service to find more documents on that topic. For example, an individual researcher

maintaining a comprehensive bibliography on some topic could submit the PubMed IDs from the bibliography to the classifier, and so easily obtain articles that the classifier judges to be relevant. As mentioned previously, with only a few relevant documents a PubMed related article search with each of the documents is probably more effective than attempting to train a classifier.

The classification service can also be used iteratively, unlike Donaldson et al. (2003) or Rubin et al. (2005) where the plan was to filter Medline once. New records can be retrieved using the date limit function on the Web form, to perform classification only on records added since the date of the last classification run. The training set could also be expanded with relevant results, and the expanded training data used to retrieve further relevant documents.

5.5 Future directions

Here we look at ways to improve the classifier and the service, either for particular circumstances such as updating results, or to improve effectiveness or speed.

Certain features would make the classifier more useful when updating results. New records are added to Medline every day, and currently one would have to periodically classify Medline while setting a date limit. It would therefore be useful to have the on-line service save the relevant training examples, and once a week re-train the classifier and run it just on the records that were added to Medline that week, then send the top-ranking results to the database curator or individual researcher. In addition, one might want to update the training examples with relevant results, and classify all of Medline again. However, the new results will have a lot of overlap with the old ones, including all the irrelevant documents that the curator viewed in the earlier results when looking for relevant documents. It would therefore be useful for the function to return only records that were not returned by the earlier classification run.

The present implementation is also missing some records: it only includes final Medline records with assigned MeSH terms, whose status is "MEDLINE". However, weeks or months may pass before a record which first appears in the daily Medline updates with "In-Progress" status re-appears with "MEDLINE" status. This means that the classifier

service is always about a month behind in journal issues. In-progress records were not included initially because of the lack of MeSH terms which were needed for the MQI feature space. Additionally, while the feature vector database can be updated with the feature vector of the finished record, the feature stream cannot. One solution would be to use the in-progress records for the weekly updates function mentioned above. This way, the main classifier results can be supplemented with a separate set of results obtained from only the in-progress records. Each week, the user would then be sent classifier results from in-progress records that appeared during the week.

The classifier algorithm itself could be made more effective, at the cost of some speed. In the training step, the rest-of-Medline is presented as irrelevant documents for training, but really constitutes unlabelled data containing some relevant documents, causing the classifier to under-fit slightly. The method of Nigam et al. (2000), Nigam (2001) incorporates unlabelled data into Bayes classifiers using expectation-maximisation (EM) to find locally maximal a posteriori models from all available data, labelled and unlabelled. Although EM requires iteration to convergence and is thus is too slow for our purposes, such a method could improve performance when relevant articles are sufficiently prevalent in the rest of Medline to adversely affect training. The present classifier also uses a multivariate Bernoulli document model for all feature spaces, including the abstract text where terms might occur more than once. At the cost of storing the number of occurrences of each term in a record, the classifier could implement a multinomial or multivariate Poisson model for abstract features only, allowing multiple occurrences of the same term to count toward classifications. Once the model accounts for term frequency, combining synonymous features could help to improve performance by better estimating their frequency.

Also, naïve Bayes classifiers, as mentioned in Section 2.2.4, are not necessarily good at ranking, because they tend to exaggerate document scores. Staging, similar to that used in Wilbur (2000), could help to improve the ranking of results. The classifier here would be the first stage, returning positive-scoring documents. A second-stage classifier would then be trained, using the same relevant training examples as the first stage, but with the positive-scoring documents returned by the first classifier used as *irrelevant* training data. Once trained, the job of the second-stage classifier is to re-rank the results of the first stage. Because the second-stage classifier only has to deal with a few thousand documents for training and ranking, a classifier with a slower training step like SVM or slower classification step like a kernel SVM could be used. The second stage may better rank relevant documents

than naïve Bayes, placing the most valuable of the relevant documents higher in the ranks.

Lastly, the efficiency of the classifier can be improved by running it in parallel. Presently, its speed is limited by the time the C program takes to read the 1.49GB feature stream from disk — although this too could be sped up by forcing a dedicated server to keep the file cached in memory. Processing the feature stream in parallel would involve splitting the file across a number of compute nodes. Each node would calculate document scores on its segment of the feature stream, and return positive-scoring PubMed IDs to the master process, which would collate them into a single ranked list and print the results. With 10 nodes, it could reduce the classification step from 168 seconds in Table 4.6 to perhaps 20 seconds. The total time to return results would be under a minute, with the sequential steps of training the classifier and formatting output dominating the running time (18-40 seconds depending on the task). Classifiers have to examine every document in the database, so on large data sets they can never be quite as fast as query-based information retrieval, where the index tables allow just documents which use query words to be examined.

Chapter 6

Concluding Remarks

We have created a classifier of Medline records, designed to identify relevant documents in Medline for the curators of biomedical databases. The classifier uses the well-established naïve Bayes approach with a multivariate Bernoulli document model, indexes the title/abstract, MeSH descriptors and qualifiers, author list and journal ISSN from Medline records, and uses an information gain criterion to select predictive features. Several characteristics distinguish the classifier from previous uses of supervised learning to filter Medline. First, the formulation of the classifier has been optimised under cross-validation on four different topics with widely varying characteristics, making it more optimised for Medline records in general, instead of being specialised to a particular subject domain. Second, the classifier judges the relevance of each of the nearly 17 million records in Medline, instead of requiring a small subset of Medline to be specified for operation. Third, only relevant training documents are specified in the input, instead of requiring a gold standard training corpus of both relevant and irrelevant documents, which may not be readily available. Fourth, we created an on-line service for the classifier that makes it simple to re-use for different classification tasks, as opposed to developing a Medline classifier solely for a particular database.

The problem of classifying all of Medline led to several innovations in formulating a classifier for large, highly skewed data sets. In operation, the classifier uses all of Medline (less submitted relevant examples) to approximate the irrelevant training data, which is made possible by the low prevalence of relevant documents for most topics. Using unlabelled

records provides a large amount of information about the term distribution and frees the classifier from requiring manually-labelled irrelevant examples in its input. Also, the naïve Bayes parameters are smoothed using a novel modification of Laplace smoothing that we call split-Laplace smoothing, which distributes the smoothing counts between the classes in proportion to their prevalence, to correct the bias of Laplace-smoothed feature weights toward the less prevalent class. Split-Laplace smoothing greatly improves cross-validation performance when the training data is highly skewed, and has the surprising property that performance does not further improve when feature selection is introduced. The gain ratio modification to information gain for feature selection was also used to compensated for class skew, enabling the same threshold to be used on differently-skewed data sets. Lastly, we used several techniques that vastly increase the efficiency of the naïve Bayes classifier — at the cost of a more complex implementation — by incorporating pre-processing steps and data structures inspired by information retrieval into a supervised learning framework.

We selected the classifier variant having the best overall cross-validation performance on four different initial corpora, and explored reasons for the performance differences between different classifier variants and between different initial corpora. Under cross-validation, the performance of the final classifier improved overall on our earlier published work. The final classifier also matches the effectiveness of an existing special-purpose database curation classifier, whose formulation had been optimised for its particular subject domain. When evaluated on test data having the same composition as Medline, the classifier retrieved two-thirds of the relevant documents found by a knowledge-engineering filter, for the same result limit, when the classifier was trained with older (non-test) relevant documents found by the knowledge-engineering filter. Ranking of Medline by classifier score also results in high precision in the upper ranks, gradually decreasing over thousands of results. The classifier is also highly efficient, taking just 168 seconds to classify all 16.6 million Medline records. The on-line service takes a little over 3 minutes from submission of the classification task to returning the result pages, which can be reduced to 69 seconds by using mainly MeSH terms for classification (discarding the title/abstract and author information).

This work removes barriers to using supervised learning on Medline. In the past, curators have had no choice but to restrict Medline to a subset, manually label training data, develop and tune a custom-built classification pipeline, and finally apply the classifier to Medline records and review the results. With this classifier, supervised learning becomes as simple as collecting existing relevant examples, submitting them to the on-line service, and

reviewing the results. The simplicity of the service makes new applications of supervised learning possible, such as an individual researcher using it to maintain a comprehensive bibliography on some topic. The classifier is open-source, and has room for improvement, for example by providing weekly updates from recently-added Medline records, incorporating term frequency information and clustering of features, improving the training on unlabelled instances in Medline, and refining the results through a second-stage classifier.

Bibliography

C Alfarano, C E Andrade, K Anthony, N Bahroos, M Bajec, K Bantoft, D Betel, B Bobechko, K Boutilier, E Burgess, K Buzadzija, R Cavero, C D'Abreo, I Donaldson, D Dorairajoo, M J Dumontier, M R Dumontier, V Earles, R Farrall, H Feldman, E Garderman, Y Gong, R Gonzaga, V Grytsan, E Gryz, V Gu, E Haldorsen, A Halupa, R Haw, A Hrvojic, L Hurrell, R Isserlin, F Jack, F Juma, A Khan, T Kon, S Konopinsky, V Le, E Lee, S Ling, M Magidin, J Moniakis, J Montojo, S Moore, B Muskat, I Ng, J P Paraiso, B Parker, G Pintilie, R Pirone, J J Salama, S Sgro, T Shan, Y Shu, J Siew, D Skinner, K Snyder, R Stasiuk, D Strumpf, B Tuekam, S Tao, Z Wang, M White, R Willis, C Wolting, S Wong, A Wrong, C Xin, R Yao, B Yates, S Zhang, K Zheng, T Pawson, B F F Ouellette, and C W V Hogue. The Biomolecular Interaction Network Database and related tools 2005 update. *Nucleic Acids Research*, 33(Database issue): D418–24, 2005. doi: 10.1093/nar/gki051. 42

Yindalon Aphinyanaphongs, Ioannis Tsamardinos, Alexander Statnikov, Douglas Hardin, and Constantin F Aliferis. Text categorization models for high-quality article retrieval in internal medicine. *Journal of the American Medical Informatics Association*, 12(2): 207–216, 2005. doi: 10.1197/jamia.M1641. 18, 24, 26, 27, 29, 31, 32, 34, 35, 43, 111, 115

J M Aronis, G F Cooper, M Kayaalp, and B G Buchanan. Identifying patient subgroups with simple Bayes. In *AMIA Annual Symposium Proceedings*, pages 658–662, 1999. 35, 43

A. R. Aronson, O. Bodenreider, H. F. Chang, S. M. Humphrey, J. G. Mork, S. J. Nelson, T. C. Rindflesch, and W. J. Wilbur. The NLM Indexing Initiative. In *AMIA Annual Symposium Proceedings*, pages 17–21, 2000. 13, 34

Alan R Aronson, James G Mork, Clifford W Gay, Susanne M Humphrey, and Willie J Rogers. The NLM Indexing Initiative's Medical Text Indexer. *Medinfo*, 11(Pt 1):268–272, 2004. 13, 34

W. C. Bartling, T. K. Schleyer, and S. Visweswaran. Retrieval and classification of dental research articles. *Advances in Dental Research*, 17:115–120, December 2003. 34, 35, 43

BMC 2008. Biomed central - the open access publisher. Online, 2008. URL `http://www.biomedcentral.com/`. 32

Andrew P. Bradley. Use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997. 26

T M Braun and T A Alonzo. A modified sign test for comparing paired roc curves. *Biostatistics*, 9(2):364–372, Apr 2008. doi: 10.1093/biostatistics/kxm036. 28, 111

Jeffrey T Chang and Russ B Altman. Extracting and characterizing gene-drug relationships from the literature. *Pharmacogenetics*, 14(9):577–586, September 2004. 40

Jeffrey T Chang, Hinrich Schütze, and Russ B Altman. GAPSCORE: finding gene and protein names one word at a time. *Bioinformatics*, 20(2):216–225, January 2004. doi: 10.1093/bioinformatics/btg393. 40

Cheetah 2008. Cheetah: The Python-powered template engine. Online, 2008. URL `http://www.cheetahtemplate.org`. 62, 66

David Chen, Hans-Michael Müller, and Paul W Sternberg. Automatic document classification of biological literature. *BMC Bioinformatics*, 7:370, 2006. doi: 10.1186/1471-2105-7-370. 14

Hao Chen and Burt M Sharp. Content-rich biological network constructed by mining PubMed abstracts. *BMC Bioinformatics*, 5(147):147, October 2004. doi: 10.1186/1471-2105-5-147. 11

Aaron M Cohen. An effective general purpose approach for automated biomedical document classification. In *AMIA Annual Symposium Proceedings*, pages 161–165, 2006. 31, 41

Aaron M Cohen and William R Hersh. A survey of current work in biomedical text mining. *Briefings in Bioinformatics*, 6(1):57–71, March 2005. doi: 10.1093/bib/6.1.57. 39

Gene Ontology Consortium. The Gene Ontology project in 2008. *Nucleic Acids Research*, 36(Database issue):D440–4, 2008. doi: 10.1093/nar/gkm883. 10, 41

William S. Cooper. Some inconsistencies and misidentified modeling assumptions in probabilistic information retrieval. *ACM Transactions in Information Systems*, 13(1):100–111, 1995. ISSN 1046-8188. doi: 10.1145/195705.195735. 19

Ido Dagan, Yael Karov, and Dan Roth. Mistake-driven learning in text categorization. *EMNLP-2: Proceedings of Second Conference on Empirical Methods in Natural Language Processing*, 1997. 15

Jesse Davis and Mark Goadrich. The relationship between precision-recall and ROC curves. In *ICML 2006: Proceedings of the 23rd International Conference on Machine learning*, pages 233–240, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143874. 27

S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407, 1990. 36

Pedro Domingos and Michael J. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997. 19

A Doms and M Schroeder. GoPubMed: exploring PubMed with the Gene Ontology. *Nucleic Acids Research*, 33(Web Server issue):783–786, July 2005. doi: 10.1093/nar/gki470. 11

Ian Donaldson, Joel Martin, Berry de Bruijn, Cheryl Wolting, Vicki Lay, Brigitte Tuekam, Shudong Zhang, Berivan Baskin, Gary D Bader, Katerina Michalickova, Tony Pawson, and Christopher W V Hogue. PreBIND and Textomy–mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics*, 4:11, March 2003. doi: 10.1186/1471-2105-4-11. 2, 18, 42, 44, 45, 46, 47, 111, 116

Chris Drummond and Robert C. Holte. Cost curves: An improved method for visualizing classifier performance. *Mach. Learn.*, 65(1):95–130, 2006. ISSN 0885-6125. doi: 10.1007/s10994-006-8199-5. 27

Stephany Duda, Constantin Aliferis, Randolph Miller, Alexander Statnikov, and Kevin Johnson. Extracting drug-drug interaction articles from MEDLINE to improve the content of drug databases. In *AMIA Annual Symposium Proceedings*, pages 216–220, 2005. 24, 26

A D Eaton. HubMed: a web-based biomedical literature search interface. *Nucleic Acids Research*, 34(Web Server issue):745–747, July 2006. doi: 10.1093/nar/gkl037. 9, 44

ElementTree 2008. Elementtree overview: sitting under the elementtree, one can feel the zen of xml. Online. URL `http://effbot.org/zone/element-index.htm`. 57

Janan T Eppig, Carol J Bult, James A Kadin, Joel E Richardson, Judith A Blake, and the members of the Mouse Genome Database Group. The Mouse Genome Database (MGD): from genes to mice–a community resource for mouse biology. *Nucleic Acids Research*, 33(Database issue):D471–D475, January 2005. 2, 41

Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. ISSN 0167-8655. doi: 10.1016/j.patrec.2005.10.010. 25, 26

George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, March 2003. URL `http://www.jmlr.org/papers/v3/forman03a.html`. 37, 38, 109

George Forman. A pitfall and solution in multi-class feature selection for text classification. In Carla E. Brodley, editor, *ICML 2004: Proceedings of the 21st International Conference on Machine Learning*, Banff, CA, 2004. Morgan Kaufmann Publishers, San Francisco, US. doi: 10.1145/1015330.1015356. 38

E Frank, M Hall, L Trigg, G Holmes, and I H Witten. Data mining in bioinformatics using WEKA. *Bioinformatics*, 20(15):2479–2481, October 2004. doi: 10.1093/bioinformatics/bth261. 2, 39

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997. ISSN 0885-6125. 19, 32

Norbert Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35 (3):243–255, 1992. ISSN 0010-4620. doi: 10.1093/comjnl/35.3.243. 17, 20

Clifford W Gay, Mehmet Kayaalp, and Alan R Aronson. Semi-automatic indexing of full text biomedical articles. In *AMIA Annual Symposium Proceedings*, pages 271–275, 2005. 13, 34

Thomas Goetz and Claus-Wilhelm von der Lieth. PubFinder: a tool for improving retrieval rate of relevant PubMed abstracts. *Nucleic Acids Research*, 33(Web Server issue):W774–W778, July 2005. doi: 10.1093/nar/gki429. 3, 43, 45, 103, 111

Google 2008. Google Scholar. Online, January 2008. URL `http://scholar.google.com`. 9

O. Gospodnetic and E. Hatcher. *Lucene in Action*. Manning Publications, Greenwich, 2005. 10, 30

J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, April 1982. 26

Feng He and Xiaoqing Ding. *Improving Naive Bayes Text Classifier Using Smoothing Methods*, pages 703–707. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007. ISBN 978-3-540-71494-1. doi: 10.1007/978-3-540-71496-5_73. 21

Healthline 2008. Healthline - health search engine and medical information. Online, 2008. URL `http://www.healthline.com`. 9

Marti A Hearst, Anna Divoli, Harendra Guturu, Alex Ksikes, Preslav Nakov, Michael A Wooldridge, and Jerry Ye. BioText search engine: beyond abstract search. *Bioinformatics*, 23(16):2196–2197, August 2007. doi: 10.1093/bioinformatics/btm301. 11

W. Hersh, A. Cohen, J. Yang, R.T. Bhupatiraju, P. Roberts, and M. Hearst. TREC 2005 genomics track overview. In *TREC 2005: Proceedings of the Fourteenth Text REtrieval Conference*, 2005. 2, 29, 41, 111

William Hersh, Chris Buckley, T. J. Leone, and David Hickam. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *SIGIR 1994: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–201, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X. 31, 111

William R. Hersh and David H. Hickam. How well do physicians use electronic information retrieval systems?: A framework for investigation and systematic review. *Journal of the American Medical Association*, 280(15):1347–1352, October 1998. doi: 10.1001/jama.280.15.1347. 8

Micheal Hewett, Diane E Oliver, Daniel L Rubin, Katrina L Easton, Joshua M Stuart, Russ B Altman, and Teri E Klein. PharmGKB: the Pharmacogenetics Knowledge Base. *Nucleic Acids Research*, 30(1):163–165, January 2002. doi: 10.1093/nar/30.1.163. 2, 40, 68

Lars J. Jensen, Jasmin Saric, and Peer Bork. Literature mining for the biologist: from information retrieval to biological discovery. *Nature Reviews Genetics*, 7(2):119–129, 2006. ISSN 1471-0056. doi: 10.1038/nrg1768. 27

Liangxiao Jiang and Harry Zhang. Learning instance greedily cloning naive bayes for ranking. In *ICDM 2005: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 202–209, Los Alamitos, CA, USA, 2005. IEEE Computer Society. doi: 10.1109/ICDM.2005.87. 18

Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *ICML 1997: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-486-3. 100

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML 1998: Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, London, UK, 1998. Springer-Verlag. ISBN 3-540-64417-2. 2, 15

Thorsten Joachims. A statistical learning model of text classification with support vector machines. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *SIGIR 2001: Proceedings of the 24th ACM International Conference on Research and Development in Information Retrieval*, pages 128–136, New Orleans, US, 2001. ACM Press, New York, US. 15

Alfons Juan, David Vilar, and Hermann Ney. Bridging the gap between naive Bayes and maximum entropy text classification. In *PRIS 2007: Proceedings of the 7th International Workshop on Pattern Recognition in Information Systems*, 2007. 17

Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. Some effective techniques for naive Bayes text classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1457–1466, 2006. ISSN 1041-4347. doi: 10.1109/TKDE.2006.180. 18, 19, 42

Ronald N Kostoff, Joel A Block, Jesse A Stump, and Kirstin M Pfeil. Information content in Medline record fields. *International Journal of Medical Informatics*, 73(6):515–527, June 2004. doi: 10.1016/j.ijmedinf.2004.02.008. 34, 35

M Lee, W Wang, and H Yu. Exploring supervised and unsupervised methods to detect topics in biomedical text. *BMC Bioinformatics*, 7:140–140, 2006. doi: 10.1186/1471-2105-7-140. 34

David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR 1992: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50, New York, NY, USA, 1992. ACM. ISBN 0-89791-523-2. doi: 10.1145/133160.133172. 30, 36

David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *ECML 1998: Proceedings of the 10th European Conference on Machine Learning*, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. 17, 18, 19, 20, 21, 48, 50

David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–306, New York, NY, USA, 1996. ACM. ISBN 0-89791-792-8. doi: 10.1145/243199.243277. 15

James Lewis, Stephan Ossowski, Justin Hicks, Mounir Errami, and Harold R Garner. Text similarity: an alternative way to search MEDLINE. *Bioinformatics*, 22(18):2298–2304, September 2006. doi: 10.1093/bioinformatics/btl388. 10

Jimmy Lin and W. John Wilbur. Pubmed related articles: a probabilistic topic-based model for content similarity. *BMC Bioinformatics*, 8(1):423, October 2007. doi: 10.1186/1471-2105-8-423. 1, 9, 10, 44, 107

LingPipe 2008. LingPipe: a suite of Java libraries for the linguistic analysis of human language. Online, 2008. URL `http://alias-i.com/lingpipe/`. 30

X. Liu and R. B. Altman. Updating a bibliography using the related articles function within PubMed. *Proceedings of the American Medical Informatics Association Symposium*, pages 750–754, 1998. 2, 44

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. 25, 58, 66

Wenlei Mao and Wesley W. Chu. The phrase-based vector space model for automatic retrieval of free-text medical documents. *Data Knowl. Eng.*, 61(1):76–92, 2007. ISSN 0169-023X. doi: 10.1016/j.datak.2006.02.008. 9

E. M. Marcotte, I. Xenarios, and D. Eisenberg. Mining literature for protein-protein interactions. *Bioinformatics*, 17(4):359–363, April 2001. doi: 10.1093/bioinformatics/17.4.359. 42, 43, 45, 111

Andrew McCallum and Kamal Nigam. A comparison of event models for naive Bayes text classification. Technical report, Just Research, 1998. 3, 16, 17, 19, 21, 37

Medscape 2008. Medscape from WebMD - search Medscape, eMedicine, MEDLINE and Drug Reference. Online, 2008. URL `http://www.medscape.com/home`. 9

Donald Metzler and W. Bruce Croft. A Markov random field model for term dependencies. In *SIGIR 2005: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. doi: 10.1145/1076034.1076115. 19, 30, 32

Tom Mitchell. *Machine Learning*. Boston: WCB/McGraw-Hill, 1997. ISBN 0-07-115467-1. 16, 18, 37

D. Mladenic and M. Grobelnik. Feature selection for unbalanced class distribution and naive bayes. In *ICML 1999: Proceedings of the Sixteenth International Conference on Machine Learning*, 1999. 38, 70, 81

Hans-Michael Müller, Eimear E Kenny, and Paul W Sternberg. Textpresso: an ontology-based information retrieval and extraction system for biological literature. *PLoS Biology*, 2(11):e309, November 2004. doi: 10.1371/journal.pbio.0020309. 2

U Mudunuri, R Stephens, D Bruining, D Liu, and F J Lebeda. botXminer: mining biomedical literature with a new web-based application. *Nucleic Acids Res*, 34(Web Server issue):748–752, Jul 2006. doi: 10.1093/nar/gkl194. 9

Kamal Nigam. *Using Unlabeled Data to Improve Text Classification*. Technical report cmu-cs-01-126, Computer Science Department, Carnegie Mellon University, 2001. 17, 117

Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39 (2/3):103–134, 2000. 21, 117

NLM 2006a. Fact sheet: PubMed®: MEDLINE® retrieval on the world wide web. Online, December 2006. URL `http://www.nlm.nih.gov/pubs/factsheets/pubmed.html`. 1, 8

NLM 2006b. National Library of Medicine PubMed subset strategies. Online, June 2006. URL `http://www.nlm.nih.gov/bsd/pubmed_subsets.html`. 12

NLM 2006c. About the UMLS® resources. Online, 2006. URL `http://www.nlm.nih.gov/research/umls/about_umls.html`. 35

NLM 2007a. National Library of Medicine AIDS subset strategy. Online, March 2007. URL `http://www.nlm.nih.gov/bsd/pubmed_subsets/aids_strategy.html`. 12, 68, 84

NLM 2007b. 2008 MEDLINE®/PubMed® Baseline distribution. Online, December 2007. URL `http://www.nlm.nih.gov/bsd/licensee/2008_stats/baseline_doc.html`. 7, 53, 56, 115

NLM 2007c. National Library of Medicine bioethics subset strategy. Online, March 2007. URL `http://www.nlm.nih.gov/bsd/pubmed_subsets/bioethics_strategy.html`. 12, 68, 84

NLM 2007d. Fact sheet: MEDLINE. Online, August 2007. URL `http://www.nlm.nih.gov/pubs/factsheets/medline.html`. 1, 7

NLM 2007e. Fact sheet: Medical Subject Headings (MeSH®). Online, October 2007. URL `http://www.nlm.nih.gov/pubs/factsheets/mesh.html`. 8, 32, 34

NLM 2008a. MEDLINE®/PubMed® data element (field) descriptions. Online, 2008. URL `http://www.nlm.nih.gov/bsd/mms/medlineelements.html`. 33

NLM 2008c. Introduction to mesh - 2008. Online, 2008. URL `http://www.nlm.nih.gov/mesh/intro_preface2008.htm`. 35

Oracle 2008. Oracle berkeley db. Online, 2008. URL `http://www.oracle.com/database/berkeley-db/db/index.html`. 57

Fuchun Peng and Dale Schuurmans. Combining naive Bayes and n-gram language models for text classification. In *ECIR 2003: Advances in Information Retrieval: 25th European Conference on IR Research*, volume 2633 of *Lecture Notes in Computer Science*, pages 335–350, Berlin / Heidelberg, 2003. European Conference on IR Research, Springer. 32

Carolina Perez-Iratxeta, Nagore Astola, Francesca D Ciccarelli, Parantu K Sha, Peer Bork, and Miguel A Andrade. A protocol for the update of references to scientific literature in biological databases. *Applied Bioinformatics*, 2(3):189–191, 2003. 2, 44

Bjoern Peters, John Sidney, Phil Bourne, Huynh-Hoa Bui, Soeren Buus, Grace Doh, Ward Fleri, Mitch Kronenberg, Ralph Kubo, Ole Lund, David Nemazee, Julia V Ponomarenko, Muthu Sathiamurthy, Stephen Schoenberger, Scott Stewart, Pamela Surko, Scott Way, Steve Wilson, and Alessandro Sette. The immune epitope database and analysis resource: from vision to blueprint. *PLoS Biology*, 3(3):e91, March 2005a. doi: 10.1371/journal.pbio.0030091. 11, 39

Bjoern Peters, John Sidney, Phil Bourne, Huynh-Hoa Bui, Soeren Buus, Grace Doh, Ward Fleri, Mitch Kronenberg, Ralph Kubo, Ole Lund, David Nemazee, Julia V Ponomarenko, Muthu Sathiamurthy, Stephen P Schoenberger, Scott Stewart, Pamela Surko, Scott Way, Steve Wilson, and Alessandro Sette. The design and implementation of the immune epitope database and analysis resource. *Immunogenetics*, 57(5):326–336, 2005b. doi: 10.1007/s00251-005-0803-5. 39

PMC 2008. PubMed Central overview. Online, January 2008. URL `http://www.pubmedcentral.nih.gov/about/intro.html`. 11, 32

M F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980. URL `http://tartarus.org/martin/PorterStemmer/`. 31

M F Porter. Snowball: A language for stemming algorithms. Online, 2001. URL `http://snowball.tartarus.org/texts/introduction.html`. 31

G L Poulter, D L Rubin, R B Altman, and C Seoighe. MScanner: a classifier for retrieving Medline citations. *BMC Bioinformatics*, 9(1):108–108, Feb 2008. doi: 10.1186/1471-2105-9-108. 5, 52, 53, 70, 72, 77, 84, 85, 86, 88, 90, 102, 108, 112, 114

J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986. doi: 10.1023/A:1022643204877. 59, 108

Dietrich Rebholz-Schuhmann, Harald Kirsch, Miguel Arregui, Sylvain Gaudan, Mark Riethoven, and Peter Stoehr. EBIMed — text crunching to gather facts for proteins from Medline. *Bioinformatics*, 23(2):e237–e244, January 2007. doi: 10.1093/bioinformatics/btl302. 1, 11

Jason Rennie, Lawrence Shih, Jaime Teevan, and David Karger. Tackling the poor assumptions of naive Bayes text classifiers. In *ICML 2003: Proceedings of the 20th International Conference on Machine Learning*, Washington, DC, 2003. Morgan Kaufmann Publishers, San Francisco, US. 20

Ellen Riloff. Little words can make a big difference for text classification. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *SIGIR 1995: Proceedings of the 18th ACM International Conference on Research and Development in Information Retrieval*, pages 130–136, Seattle, US, 1995. ACM Press, New York, US. 31, 37

Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC-3. In *TREC-3: Proceedings of the 3rd Text REtrieval Conference*, pages 21–30, 1992. 10

J. J. Rocchio. *Relevance feedback in information retrieval*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, USA, 1971. 15

Daniel L Rubin, Michelle Carrillo, Mark Woon, John Conroy, Teri E Klein, and Russ B Altman. A resource to acquire and summarize pharmacogenetics knowledge in the literature. *Medinfo*, 11(Pt 2):793–797, 2004. 40

Daniel L Rubin, Caroline F Thorn, Teri E Klein, and Russ B Altman. A statistical approach to scanning the biomedical literature for pharmacogenetics knowledge. *Journal of the American Medical Informatics Association*, 12(2):121–129, 2005. doi: 10.1197/jamia.M1640. 3, 19, 35, 40, 42, 44, 45, 46, 104, 111, 115, 116

Gerard Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*, chapter 10. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989. ISBN 0-201-12227-8. 9, 10, 15

Martijn J Schuemie and Jan A Kors. Jane: Suggesting journals, finding experts. *Bioinformatics*, 2008. doi: 10.1093/bioinformatics/btn006. 10

Fabrizio Sebastiani. A tutorial on automated text categorisation. In Analia Amandi and Ricardo Zunino, editors, *ASAI 1999: Proceedings of the 1st Argentinian Symposium on Artificial Intelligence*, pages 7–35, Buenos Aires, AR, 1999. 30

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002. ISSN 0360-0300. doi: 10.1145/505282.505283. 2, 12, 13, 18, 22, 23, 24, 31, 32, 36, 37, 38

Mir S Siadaty, Jianfen Shu, and William A Knaus. Relemed: sentence-level search engine with relevance score for the MEDLINE database of biomedical articles. *BMC Medical Informatics and Decision Making*, 7:1, 2007. doi: 10.1186/1472-6947-7-1. 1, 9

SQLite 2008. Sqlite is a software library that implements a self-contained, serverless, zero-configuration, transactional sql database engine. Online, 2008. URL `http://www.sqlite.org`. 57

Brian P Suomela and Miguel A Andrade. Ranking the whole MEDLINE database according to a large training set using text indexing. *BMC Bioinformatics*, 6:75, 2005. doi: 10.1186/1471-2105-6-75. 3, 45, 111

Imad Tbahriti, Christine Chichester, Frédérique Lisacek, and Patrick Ruch. Using argumentation to retrieve articles with similar citations: an inquiry into improving related articles search in the MEDLINE digital library. *International Journal of Medical Informatics*, 75(6):488–495, June 2005. doi: 10.1016/j.ijmedinf.2005.06.007. 10

Guido van Rossum and Fred L. Drake. *Python Reference Manual*. Virginia, USA, 2001. URL `http://www.python.org`. 60

D Vilar, H Ney, A Juan, and E Vidal. Effect of feature smoothing methods in text classification tasks. In *PRIS 2004: Proceedings of the 4th International Workshop on Pattern Recognition in Information Systems*, pages 108–117, April 2004. 21

Peng Wang, Alexander A Morgan, Qing Zhang, Alessandro Sette, and Bjoern Peters. Automating document classification for the Immune Epitope Database. *BMC Bioinformatics*, 8:269, 2007. doi: 10.1186/1471-2105-8-269. v, 2, 3, 5, 11, 12, 19, 26, 27, 29, 31, 32, 34, 35, 37, 39, 45, 47, 59, 67, 68, 69, 72, 82, 85, 90, 91, 103, 104, 110, 111, 112, 115

WebPy 2008. Web.py: Think about the ideal way to write a web app. write the code to make it happen. Online, 2008. URL `http://webpy.org`. 62

W. J. Wilbur. Boosting naïve Bayesian learning on a large subset of MEDLINE. *Proceedings of the American Medical Informatics Association Symposium*, pages 918–922, 2000. 22, 31, 43, 117

Nancy L Wilczynski, Douglas Morgan, and R Brian Haynes. An overview of the design and methods for retrieving high-quality studies for clinical care. *BMC Medical Informatics and Decision Making*, 5:20, 2005. doi: 10.1186/1472-6947-5-20. 12

Ioannis Xenarios, Lukasz Salwínski, Xiaoqun Joyce Duan, Patrick Higney, Sul-Min Kim, and David Eisenberg. DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, 30(1):303–305, January 2002. 42

Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR 1999: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, Berkley, August 1999. 18, 29

Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *ICML 1997: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US. 36, 37, 38, 81, 109

Meliha Yetisgen-Yildiz and Wanda Pratt. The effect of feature representation on MEDLINE document classification. In *AMIA Annual Symposium Proceedings*, pages 849–853, 2005. 31

Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. Feature selection for text categorization on imbalanced data. *SIGKDD Explorations*, 6(1):80–89, 2004. doi: 10.1145/1007730.1007741. 38, 59