Automatic Identification of Protein Characterization Articles in support of Database Curation

by

Robert E. Denroche

A thesis submitted to the School of Computing
In conformity with the requirements for
the degree of Masters of Science

Queen's University
Kingston, Ontario, Canada
(January, 2010)

Abstract

Experimentally determining the biological function of a protein is a process known as protein characterization. Establishing the role a specific protein plays is a vital step toward fully understanding the biochemical processes that drive life in all its forms. In order for researchers to efficiently locate and benefit from the results of protein characterization experiments, the relevant information is compiled into public databases. To populate such databases, curators, who are experts in the biomedical domain, must search the literature to obtain the relevant information, as the experiment results are typically published in scientific journals. The database curators identify relevant journal articles, read them, and then extract the required information into the database. In recent years the rate of biomedical research has greatly increased, and database curators are unable to keep pace with the number of articles being published. Consequently, maintaining an up-to-date database of characterized proteins, let alone populating a new database, has become a daunting task.

In this thesis, we report our work to reduce the effort required from database curators in order to create and maintain a database of characterized proteins. We describe a system we have designed for automatically identifying relevant articles that discuss the results of protein characterization experiments. Classifiers are trained and tested using a large dataset of abstracts, which we collected from articles referenced in public databases, as well as small datasets of manually labeled abstracts. We evaluate both a standard and a modified naïve Bayes classifier and examine several different feature sets for representing articles. Our findings indicate that the resulting classifier performs well enough to be considered useful by the curators of a characterized protein database.

Co-Authorship

Portions of the information and results discussed in Chapter 3, in Chapter 5 (excluding Section 5.1) and in Chapter 6 were presented at ISMB 2009 BioLINK SIG and published in the following paper:

R. Denroche, R. Madupu, S. Yooseph, G. Sutton and H. Shatkay. Towards Computer-Assisted Text Curation: Classification is Easy (Choosing Training Data can be Hard…). *Proceedings of the ISMB'09 BioLINK SIG, Springer Lecture Notes in Bioinformatics*, to appear, 2010.

Acknowledgements

First and foremost, I'd like to express my gratitude towards my supervisor, Dr. Hagit Shatkay. Without her insight and guidance I would not have had the opportunity to pursue such fulfilling research nor the ability to complete it.

I would like to thank my defence committee, Dr. Paul Young, Dr. James Stewart and Dr. Robin Dawes, for their helpful questions and comments.

I would also like to thank my labmates, Sara, Na, Phil, Yin and Andrew, for the always pleasant discussions and for lab lunches.

I would like to express my gratefulness towards the Queen's University School of Computing for funding my research and for harboring such wonderful faculty, staff and students. Over the course of my Bachelor's and Master's degrees, they have imparted me with knowledge I will always value and memories I will forever cherish.

I'd like to express my thanks to the J. Craig Venter Institute for also funding my research. My collaborators at the JCVI, Dr. Ramana Madupu, Dr. Shibu Yooseph, and Dr. Granger Sutton, were always approachable and a great source of knowledge and direction. Special thanks go to Ramana for her work labeling the articles in our curated datasets. I would also like to thank my fellow interns at the JCVI, for being friendly and for after-work Frisbee.

Heartfelt thanks go out to my family, for their continuous support over the years, and for driving me to the rink at 6 am.

Finally, I'd like to thank Elyse for her love.

Table of Contents

Abstract ii
Co-Authorship iii
Acknowledgements iv
Table of Contents v
List of Figures vii
List of Tables viii
Chapter 1 Introduction 1
1.1 Motivation 2
1.2 Target Application 3
1.3 Contributions 4
1.4 Thesis Organization 5
Chapter 2 Background 6
2.1 Genome Annotation and Protein Characterization 6
2.1.1 Genome Annotation 7
2.1.2 Protein Characterization 9
2.2 JCVI Annotation Pipeline and the CHAR Database 10
2.3 Introduction to Text Classification 12
2.3.1 Text Representation 12
2.3.2 Feature Selection 14
2.3.3 Text Categorization 17
2.3.4 Performance Evaluation 19
2.4 Related Work 21
Chapter 3 Dataset Construction 29
3.1 The ***Curated Journal*** Dataset 30
3.2 The ***Curated Swiss-Prot*** Dataset 31
3.3 The ***SP-GO*** Dataset 32
3.4 The ***CHAR*** Dataset 39
Chapter 4 Text Preparation and Classification 42
4.1 Text Preprocessing 42
4.2 Feature Selection 44
4.2.1 Feature selection using Document Frequency Thresholding 45
4.2.2 Feature selection using *Z-score* 46
4.2.3 Replacing or removing Species-Specific Terms 48
4.3 Article Representation 51
4.4 Classification 52
4.4.1 Standard Naïve Bayes Classifier 53

4.4.2 Only-present-terms Naïve Bayes Modification 56
Chapter 5 Results 58
5.1 Experiment Design 58
5.1.1 Feature Selection 60
5.1.2 Standard and Only-Present-Terms Naïve Bayes Classifiers 62
5.1.3 Feature Set Optimization 62
5.2 Results for Classifiers trained on the ***SP-GO*** Dataset 64
5.2.1 Cross-Validation Results for Classifiers trained on the ***SP-GO*** Dataset 65
5.2.2 Test Results for Classifiers trained on the ***SP-GO*** Dataset 67
5.3 Results for Classifiers trained on the Curated Datasets 69
5.3.1 Results for Classifiers trained on the ***Curated Journal*** Dataset 70
5.3.2 Results for Classifiers trained on the ***Curated Swiss-Prot*** Dataset 71
5.4 Kullback-Leibler Divergence Analysis of Term Distributions 72
5.4.1 KL Divergence Results 73
5.4.2 Separating the ***SP-GO*** Dataset 75
5.5 Results for Classifiers trained on the ***SPonly*** Dataset 76
5.5.1 Cross-Validation Results for Classifiers trained on the ***SPonly*** Dataset 76
5.5.2 Test Results for Classifiers trained on the ***SPonly*** Dataset 80
Chapter 6 Term Distribution Analysis 84
6.1 Construction and Term Distribution of the ***SP-GO*** Dataset 85
6.2 Construction and Term Distribution of the ***SPonly*** Dataset 88
Chapter 7 Conclusions and Future Work 92
7.1 Contributions 93
7.2 Future Work 95
Bibliography 99
Appendix A Additional Biology Background 105
A.1 Automated Genome Annotation 105
A.2 Protein Characterization 107
Appendix B Full Results for Classifiers trained on the ***SP-GO*** Dataset 112
Appendix C Cross-Validation and Test Results on the Curated Datasets 127
Appendix D Full Results for Classifiers trained on the ***SPonly*** Dataset 130
Appendix E Top terms by *Z-Score* from the ***SP-GO*** and ***SPonly*** Datasets 148
Appendix F Implementation Notes 151

List of Figures

Figure 3.1: PubMed IDs for the *positive* and *negative* articles in the initial ***SP-GO*** dataset 36

Figure 3.2: Cross-validation results over the initial ***SP-GO*** dataset 37

Figure 3.3: PubMed IDs for the *positive* and *negative* articles in the final ***SP-GO*** dataset 40

Figure 3.4: Cross-validation results over the final ***SP-GO*** dataset 41

Figure 4.1: A sample of species name entries from NCBI's Taxonomy Database 51

Figure D.1: Classification results at several *Z-score* thresholds for classifiers trained and tested on the ***SPonly*** dataset using cross-validation 133

Figure D.2: Accuracy at several *Z-score* thresholds for classifiers trained on the ***SPonly*** dataset and tested on the other datasets..... 140

List of Tables

Table 3.1: Curation labels for the ***Curated Journal*** dataset 30

Table 3.2: Curation labels for the ***Curated Swiss-Prot*** dataset 31

Table 5.1: Default Feature Selection Parameters 64

Table 5.2: Feature Selection Parameters evaluated with Classifiers trained on the ***SP-GO*** Dataset 66

Table 5.3: Cross-validation results for the standard naïve Bayes classifier trained on the ***SP-GO*** dataset over the default feature set 66

Table 5.4: Cross-validation results for the only-present-terms naïve Bayes classifier trained on the ***SP-GO*** dataset over the default feature set 67

Table 5.5: Test results for a classifier trained using the ***SP-GO*** dataset and tested on the other datasets 68

Table 5.6: Test results for a classifier trained on the ***Curated Journal*** Dataset 71

Table 5.7: Test Results for a classifier trained on the ***Curated Swiss-Prot*** Dataset 71

Table 5.8: Kullback-Leibler Divergence between dataset pairs 74

Table 5.9: KL divergence between each of the ***SPonly*** and ***GOonly*** sets and all other subsets 75

Table 5.10: Feature Selection Parameters evaluated with Classifiers trained on the ***SPonly*** Dataset 77

Table 5.11: Cross-validation results for standard naïve Bayes classifiers trained on the ***SPonly*** dataset 78

Table 5.12: Cross-validation results for only-present-term naïve Bayes classifiers trained on the ***SPonly*** dataset 79

Table 5.13: Test results for a standard naïve Bayes classifier trained over the ***SPonly*** dataset and tested on the other datasets 81

Table 5.14: Test results for an only-present-terms naïve Bayes classifier trained using the ***SPonly*** dataset and tested on the other datasets 82

Table 6.1: IDA or IPI related terms that are overrepresented in the *positive* articles of the ***SP-GO*** dataset 87

Table 6.2: Top 5 Swiss-Prot scopes for the 1451 *negative* articles from the ***SPonly*** dataset 90

Table 6.3: Sequence-related terms that are overrepresented in the *negative* articles of the ***SPonly*** dataset 91

Chapter 1

Introduction

Protein characterization is the process of determining the biological function of a protein. Proteins perform a multitude of different biological functions such as creating or serving as structures, recognizing or transporting other molecules or facilitating chemical reactions. Determining the function of a specific protein is a labor-intensive task. The results of protein characterization experiments are published in scientific journals.

Protein information, including biological function, is collected into databases so that researchers can benefit from work that had already been published. In order to create and maintain such databases, database curators – experts with extensive experience in the biomedical domain – must locate potentially relevant journal articles and then read them in order to extract the relevant information. Manual database curation is a time-consuming process.

In order to reduce the curator effort needed to create and maintain a protein database, we developed a system capable of automatically identifying potentially relevant journal articles. We focus on locating articles that report the results of protein characterization experiments. Our datasets and experimental design were created through

collaboration with the curators of the Characterized Protein Database (CHAR) [38]. However, our findings are applicable to any task where human effort can be reduced by automatically identifying protein characterization articles, as well as more general text classification tasks.

1.1 Motivation

Proteins are sequences of amino acids. The specific sequence of amino acids required to create a protein is encoded in the sequence of the nucleotides that make up a gene. Many public databases are actively collecting the sequences of genes and proteins along with relevant information on their biological function. Species specific databases such as the FlyBase [14], the Mouse Genome Informatics Database [42], and the Saccharomyces Genome Database [58], or general repositories including GenBank [43] and Swiss-Prot [61] are constantly being updated with new gene and protein information. In order to add new information, and ensure that the information is reliable, expert curators must locate and read potentially relevant articles. Manual curation is a slow process that is necessary to ensure that the information in databases is correct.

Recent development of automated, high-throughput techniques allow researchers to quickly and efficiently determine the sequence of the nucleotides in an organism's genome. Genome annotation is the process of locating the genes in a genome sequence and then assigning useful information to them. Since genes encode proteins, one important type of information associated with each gene is the function of the protein it encodes. However, as we discussed earlier, characterizing the function of a single protein

requires months of experimentation, which is much slower than the speed of genome sequencing. Currently, instead of waiting for each new protein to be characterized experimentally, genome annotation is performed by searching databases for other proteins that are similar to the new protein. If the similar proteins have already been characterized and are sufficiently similar to the new protein to suggest that they share the same function, the new protein can be assigned the function of the similar proteins. Genome annotation would not be possible without reliable protein databases.

1.2 Target Application

The J. Craig Venter Institute (JCVI) is currently developing a highly reliable, fully curated collection of protein information known as the Characterized Protein Database (CHAR). The database only contains proteins that have been experimentally characterized, and each protein is linked to the articles that describe the protein's characterization. JCVI currently uses CHAR as a trusted source of evidence to support the annotation of the genomes that they sequence, and plans to make CHAR a publicly available resource once it has been fully developed and populated.

In order to add new proteins to the Characterized Protein database, curators must search for scientific journal articles that are potentially relevant to CHAR, read them to determine if they contain relevant protein characterization experiments and then extract the relevant information into the database. As noted earlier, this is a slow process, involving much curator time and effort.

In this thesis, we report our work to create a system capable of classifying journal articles based on their relevance to the CHAR database. Identifying relevant articles can save curator effort, and is the first step towards a system capable of automatically extracting information from the articles and populating database entries. Because our work was motivated and supported by the J. Craig Venter Institute, we created datasets and designed our experiments with the CHAR database in mind. Furthermore, we evaluate the performance of our classifier by comparing our results to requirements specified by CHAR's curators. Despite this focus, our work is applicable to any task where curator effort can be reduced by automatically identifying protein characterization articles, and some of our findings apply to the more general aspects of biomedical text classification.

1.3 Contributions

In this thesis, we make several contributions:

- We investigate the feasibility of several feature selection approaches, and evaluate the performance of both standard and modified naïve Bayes classifiers.
- We show that building a large dataset of many articles collected based on the curator labels from public databases is not always an effective approach for a text classification task. The classifier we trained on our large dataset performed poorly, misclassifying the majority of articles on their relevance to CHAR, despite the size of the dataset and the careful consideration invested in its construction.

- In contrast, we observe that classifiers trained using small datasets of manually curated articles show much better performance. This result suggests that using small, manually curated datasets to train classifiers is likely to be a viable approach for supporting future text classification tasks.
- Finally, we demonstrate that naïve Bayes classifiers trained on a subset of our largest dataset perform at a level surpassing the requirements specified by CHAR curators over two of our three test sets.

1.4 Thesis Organization

The remainder of this thesis is organized as follows: We present an overview of the relevant biological concepts, a summary of typical text classification methods and a survey of the related literature in Chapter 2. The collections of scientific journal articles, which we use as training and test sets, as well as the methods used for their construction, are described in Chapter 3. Our text-processing techniques, feature selection and classification methods are described in Chapter 4. We report and discuss our experimental results over the various datasets in Chapter 5. We analyze our datasets in order to explain the performance of our classifiers in Chapter 6. Chapter 7 provides conclusions and an outline of future work.

Chapter 2

Background

In this chapter we provide the relevant background required to fully understand this thesis. We begin with an overview of the biology and techniques involved in genome annotation and protein characterization. Next we briefly describe the J. Craig Venter Institute's database of Characterized Proteins (CHAR) which is used in its genome annotation efforts. We then provide an introduction to text classification, including text processing and representation, feature selection, classification approaches and performance evaluation. Finally, we survey literature that describes approaches for text classification tasks similar to our own.

2.1 Genome Annotation and Protein Characterization

A genome is the complete genetic information for an organism, encoded as a sequence of nucleotides that make up the DNA (deoxyribonucleic acid) [65]. There are 4 different types of nucleotides, and their order in the DNA molecule forms the genetic sequence of the organism. An organism's genetic sequence consists of genes and the non-coding regions between the genes. Genes encode the sequence of the amino acids in a protein.

Proteins perform a myriad of different functions that are essential to the existence and survival of the organism. Some proteins combine to form large structures which serve, for instance, as the framework that maintains the cell's shape or the mechanism that allows muscle cells to contract. Other proteins are enzymes that catalyze various chemical reactions. Enzymes control the rate of digestion or construction of molecules within the cell, thus enabling the cell's survival. The structure and chemical properties of many proteins enable them to bind to other molecules with high specificity. This lets some proteins transport desired molecules into the cell or between subcellular locations, bind to specific locations on the DNA, or identify other cells as part of the immune system.

The non-coding regions of the genome are the nucleotides that lie between the genes. These sequences are not directly expressed as proteins. However, the non-coding regions can influence where proteins can bind to the DNA molecule and how the DNA molecule folds up inside the cell, potentially regulating the expression of the genes [6].

2.1.1 Genome Annotation

Genome sequencing is the process of determining the sequence of nucleotides that make up an organism's genome [65]. Due to scientific and technological advances, the pace of genome sequencing is constantly increasing, and the number of nucleotides in online databases grows at an ever increasing rate. The GenBank database [43], which is a collection of all publicly available DNA sequences, contained 105,277,306,080 nucleotides as of its June 15th, 2009 release, [44].

In order to make full use of the information contained in an organism's genome, researchers need to know where the genes are and what the genes do. As discussed in Section 1.1, genome annotation is the process of locating the genes and then labeling them with useful information, including the function of the proteins they encode. Determining the location of the genes on the genome sequence is known as *gene finding.* Segments of the genome that are likely to be genes can be located automatically by computer programs, such as GENESCAN [4] or GLIMMER [9], that use patterns or statistical models to predict where the genes are located. Genes can also be found by comparing the genome sequence to the sequences of known genes from other species; if a portion of the genome appears very similar to a known gene, it is likely that the portion is, in fact, a gene. Modern gene finding techniques often combine statistical and comparative approaches [60].

Once a gene has been located, it is often possible to determine the amino acid sequence of its protein, and then annotate the protein with its function. There are many experimental ways to infer the function of the protein, which we discuss in Section 2.1.2. However, such lab based techniques cannot keep up with the pace of high-throughput genome sequencing, which is capable of sequencing a small genome in a matter of days [40]. Computational genome annotation methods have been developed in order to quickly assign biological information to new proteins based on proteins that have already been experimentally characterized [60]. Automatic genome annotation can be performed by using the new protein's sequence to search for homologues (proteins with similar

sequence and likely similar function) and to compare the new protein to sequence models that represent families of proteins with similar structure and function [1, 20]. We describe both of these approaches in detail in Appendix A.1. In practice, systems often employ both approaches and use expert curators to confirm the resulting annotation [38].

2.1.2 Protein Characterization

Protein characterization is the process of determining the biological function of a protein. There are several ways to directly discover the function of proteins but all are intensive, lab based approaches. Currently, researchers must characterize proteins individually, through experimentation, which is a slow process. Experimental characterization results are published in scientific journals. Researchers can determine the protein's functions through several techniques, and, ideally, the function can be confirmed through additional types of experiments.

The first step of many protein characterization experiments is to isolate the protein, which is a difficult task because cells contain thousands of proteins. Once isolated, tests can be performed on the protein to determine its amino acid sequence, what molecules it reacts with or what its structure is. Other experiments involve modifying a gene so that its protein is over-expressed, or not expressed at all, and then observing the effect on the organism. Specific proteins can be made to fluoresce so that their location in the cell can be observed with a microscope. Finally, some techniques, such as *yeast two-hybrid* experiments, determine whether two proteins are capable of interacting and

binding together. We provide an overview of some of the many different types of protein characterization experiments in Appendix A.2.

2.2 JCVI Annotation Pipeline and the CHAR Database

Our research was motivated (as well as partially funded and supported) by the J. Craig Venter Institute (JCVI). JCVI is developing CHAR, a database of Characterized Proteins, to support their genome annotation pipeline. In this section we describe JCVI's annotation pipeline and the CHAR database in order to illustrate the target application for our work.

The J. Craig Venter Institute's Joint Technology Center (JTC) is a state of the art high-throughput DNA sequencing facility. JCVI has developed an automated genome annotation pipeline capable of finding the genes on the genome of prokaryotes[1], predicting the proteins that the genes encode, and then determining the function of the proteins. JCVI uses this pipeline to annotate their sequences, and also allows other researchers to run prokaryotic genomes through the pipeline free of charge [64].

The JCVI annotation pipeline uses a number of resources in order to assign a function to each protein sequence. Sequence similarity searches are performed (using BLAST [1]) against publicly available sequence databases and statistical models of protein families are compared to the new protein sequence. Some resources, including

[1] Prokaryotes are a group of organisms whose DNA is not held inside the subcellular structure known as the nucleus. All bacteria are prokaryotes.

statistical models such as TIGRFAM protein families [20], are considered very reliable by the CHAR curators. However, in other resources, including entries from protein sequence databases such as Swiss-Prot [61], it is not always clear if the protein has been experimentally characterized, which is why expert curators review the annotations created by the pipeline [38]. Curators evaluate the quality of the information associated with the proteins that are most similar to the new protein. By examining the articles referenced by the database, the curators ensure that the similar proteins have indeed been experimentally characterized, and that the associated information does not contain any errors.

The Characterized Protein Database (CHAR) aims to be a trusted and reliable resource for JCVI's genome annotation curators. Whenever a curator evaluates an entry in a public protein sequence database and determines that the entry is correct and the protein has been experimentally characterized, the curator is encouraged to create an entry for that protein in the CHAR database. This way, the next time a new protein is found to be similar to a protein whose entry has been reviewed, the reliable protein will have an entry in the CHAR database and a curator will not have to spend time reinvestigating the public database entry. Each CHAR entry contains the protein's name, sequence, biological function, a list of other public databases where the protein may be found and references to the scientific journal articles that report the experimental characterization of the protein. Once CHAR is fully developed and populated, it will be

released as a BLAST-searchable public resource. As of June 2009, the CHAR database contained almost 12,000 expert-curated entries for fully characterized proteins [38].

In order to populate the CHAR database, curators must first identify relevant journal articles and then extract the protein characterization results. We aim to aid the database curators by identifying relevant articles automatically. To this end, we designed a system capable of classifying journal articles based on their relevance to the CHAR database.

2.3 Introduction to Text Classification

Text classification is the process of taking a unit of text, be it a sentence, paragraph or full article, and assigning it to one of a predetermined set of classes. In this thesis, the text consists of the title and the abstract of scientific journal articles. We have two classes: articles that are *relevant* to the CHAR database, and articles that are *irrelevant*. *Relevant* articles contain experimental characterization of proteins, and *irrelevant* articles do not contain desired protein information. In this section, we introduce the typical steps performed when classifying text and briefly outline some techniques. We refer the reader to the thorough overview by Sebastiani for additional information [57].

2.3.1 Text Representation

Text must be converted from natural language into a machine readable vector of features before classification techniques can be applied. Text is represented as an m-dimensional vector of features, where m is the number of features. Each feature represents a term, and

the value of each feature represents the weight of the term in the text. Terms can be words or phrases from the text; there are several ways to calculate their weight in the vector.

The most straightforward text representation approach is to use single words as terms. However, the structure of the sentence and, therefore, the context is lost when only single words are considered. Two approaches that aim to retain some sentence structure, with the goal of improving classification, are to use syntactical phrases or statistical phrases as terms.

Syntactical phrases are small sequences of words that are syntactically connected in the text, forming, for example, *noun* phrases or *verb* phrases. These terms capture the syntactical relationship amongst the words. However, identifying syntax in natural language is a difficult task, and research has shown that including syntactical phrases as terms only results in small improvements of performance over approaches that only use terms created from single words [34].

Statistical phrases, also known as *n*-grams, are created by collecting sequences of *n* consecutive words into a phrase. An *n*-gram is used to capture the order in which the words occur, and it has been shown that using bigrams (where *n* is 2) and trigrams (where *n* is 3) in addition to single words can improve classification performance [16]. For example, in our work, the words yeast, two or hybrid may individually occur for a variety of reasons, but when consecutively occurring as the trigram yeast-two-hybrid,

which is a type of protein characterization experiment, they are likely to indicate that the article is relevant.

There are several methods to calculate the weight for each term. The simplest approach, known as a *binary* representation or a *multi-variate Bernoulli* event model, assigns a weight of 1 for each term that is present in the text and 0 otherwise [41]. Other approaches take into account how often each term occurs in the text, with the idea being that if some terms occur frequently in the text, those terms must be important. Under a *term frequency* (*tf*) representation, the weight for each term is calculated by counting the number of occurrences of that term in the text and dividing by the total number of terms in the text [31]. Term frequency is often normalized by the *inverse document frequency* (*idf*), resulting in the most common weighting scheme: *tf·idf*. The document frequency of a term is calculated by counting the number of documents that contain the term and dividing by the total number of documents [55]. Multiplying *tf* by *idf* reduces the relative weight of terms that are frequent in many documents and increases the relative weight of terms that occur rarely within all the documents. Many other word weighting schemes exist [31, 55], but the end result of all approaches is having the text represented by a feature vector of term weights.

2.3.2 Feature Selection

When representing text as a vector of features, the large number of features involved can render many machine learning methods ineffective due to the high-dimensionality of the training vectors. Furthermore, some features may not provide information that is

beneficial to the classification task, and using them to predict a class for the text could result in lower performance. Feature selection addresses both of these issues by reducing the number of features and removing uninformative ones.

There are a number of text processing methods that are often employed in order to reduce the number of features involved in a text classification. *Stop words* are common, topic-neutral words such as the, and or not. Since their presence is typically not related to the class of the document, they can be removed without lowering classification performance. Words can be stemmed in order to collapse them into their lexical roots (e.g. characterize, characterized and characterization could all be represented by the term character-) [49]. The result of stemming is a reduced number of features.

Additional feature selection techniques leverage statistics regarding how often terms appear in the documents of each class in order to remove non-informative terms. In *document frequency* filtering, the terms that occur in only a few or nearly all of the documents are removed. Frequent terms are often non-informative for classification because they occur frequently in most documents, regardless of the class of the documents. Likewise, rare terms are non-informative because they rarely appear in any documents, regardless of class. Neither case supports the goal of learning the difference between the documents of each class, therefore removing the rare and frequent terms reduces the number of features that need to be considered and has the potential to improve classification performance [69].

Various statistical tests can be used to rank features based on their ability to discriminate between classes (*t-statistic* [56] or *Z-statistic* [3, 15]), or on the dependence between the term and the class (χ^2*-statistic* [48, 7]). Once the features are ranked, the least informative features can be removed in order to reduce the total number of features and potentially improve performance.

Other measures that can be used to identify non-informative features include *information gain*, which measures the number of bits of information that can be obtained about the category by knowing if a term is present or absent in the document, and *term strength*, which is an estimate of the utility for each term based on how likely that term is to appear in "closely-related" documents [69].

Alternative feature selection approaches include methods that take the original feature set, where each feature corresponds to a term, and extract a new set of features. In *Principal Component Analysis* (PCA), the feature space is projected onto a lower dimensional space where each new feature is a linear combination of the original features. PCA effectively reduces the number of features and can improve the performance of a classifier [30, 36]. However, since each new feature is a combination of the original features, interpreting the results becomes difficult as the features no longer correspond to single terms. Another approach is *Latent Semantic Indexing* (LSI), where the feature space is compressed into a set of lower dimension matrices using *Singular Value Decomposition* (SVD) [8, 57]. Manipulation of the matrices can reveal relationships between the terms and the classes, and new documents can be assigned to

classes using this approach, but, again, the results can be difficult to interpret because each feature no longer corresponds to a single term.

2.3.3 Text Categorization

Text categorization is the process of taking the feature vector representation for a document and assigning it to a class. Several machine learning techniques have been applied to text categorization tasks, including *Hierarchical Clustering* [32, 72], *Maximum Entropy* [46], *K-Nearest Neighbour* (KNN) [69], *Expectation-Maximization* [47], *Support Vector Machines* (SVM) [27, 56, 11, 7], *Neural Networks* [30], *Decision Trees* [2, 36] and *Naïve Bayes classifiers* [41, 48, 54, 36, 11, 7]. The two main types of machine learning methods are *unsupervised learning* and *supervised learning*. In unsupervised learning approaches, such as Hierarchical Clustering [32, 72] or Expectation-Maximization [47], patterns are discovered in the data and the unlabeled feature vectors are grouped into different clusters. In supervised learning, however, feature vectors that have been labeled with the correct class are used as training examples so that a model or function can be learned and then applied in order to assign class labels to new, yet unseen, feature vectors. K-Nearest Neighbours, Support Vector Machines and Naïve Bayes classifiers are all examples of supervised learning approaches. We use a supervised learning technique for our task because we have a clear understanding of the classes involved (articles are either *relevant* or *irrelevant* to CHAR), and have the means to collect labeled examples to use for training. In the following paragraphs, we discuss three supervised machine learning techniques that have been applied to text classification.

K-Nearest Neighbours is a simple classification method that can be applied to text. During the learning phase, the training examples (class-labeled feature vectors) are simply stored for later reference. To classify a new feature vector, a similarity measure is computed between the new vector and all of the training vectors. The new vector is then assigned to a class based on the classes of the *k* most similar training vectors (or the *k*-nearest neighbours). KNN has proven to be an effective text classification approach [69]. However, storing and then calculating similarities to all of the training examples can be computationally expensive, which is why we do not employ this technique.

Support Vector Machines are another type of classifier that has been shown to perform well on text categorization tasks [27, 11]. SVMs work by mapping the feature vector into a higher dimensional space and then determining the hyperplane that best separates the example vectors from the two classes. Some of the possible transformations used to map the features in the higher dimensional space include linear, polynomial, radial and hyperbolic transformations. The hyperplane is positioned so as to maximize the margin between the training examples and the hyperplane, resulting in the greatest separation of the examples from each class. SVMs are effective classifiers, but they are difficult to implement, have high computational complexity and space requirements [5], and it can be difficult to determine the optimal parameters used to map the features and place the hyperplane. It is for these reasons that we do not employ a Support Vector Machine in this study.

Bayes' theorem can be used to determine which class is the most likely given a feature vector, based on statistics gathered from the training data. A Naïve Bayes classifier is based on the assumption that all features are conditionally independent of each other given the class [54, 12]. The classifier is learned by taking maximum likelihood estimates of the feature probability distributions from the training data. Then, in order to classify a new feature vector, the probability of the vector belonging to each class is computed, and the feature vector is assigned to the class that maximizes this probability. Naïve Bayes classifiers are simple to implement and have been applied to many text classification tasks, often performing well [41, 48, 54, 36, 11]. For these reasons, we use a Naïve Bayes classifier for our classification task, and thus describe it in greater detail in Section 4.4.1.

2.3.4 Performance Evaluation

Text classification systems are primarily evaluated based on the number of articles classified correctly compared to the number of articles classified incorrectly. We define our performance metrics in terms of four values: the number of *relevant* articles classified correctly (*true positives, TP*), the number of *irrelevant* articles classified correctly (*true negatives, TN*), the number of *relevant* articles classified incorrectly (*false negatives, FN*) and the number of *irrelevant* articles classified incorrectly (*false positives, FP*).

Classification *accuracy*, which is the number of correctly classified articles divided by the total number of articles, is a common measure of performance which can be used to evaluate a classifier. It is defined as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

However, accuracy can be misleading when the articles to be classified are not distributed uniformly across the classes. In many text classification applications where the decision to be made for each article is whether it is *relevant* to a given topic or not, the number of articles *relevant* to one topic can be quite small compared to the number of articles that are *irrelevant*. A classifier may obtain high accuracy by simply labeling all articles as *irrelevant*, classifying the many *irrelevant* articles correctly and mislabeling the few *relevant* ones. However, since the task is to identify the *relevant* articles, this high accuracy would not indicate the desirable performance.

Recall and *precision* are the metrics most commonly used to evaluate the performance of text classifiers [52]. They are defined as follows:

$$recall = \frac{TP}{TP + FN}, \text{ and } precision = \frac{TP}{TP + FP}.$$

Both these measures focus on the performance over the *relevant* articles, which makes them more informative than accuracy when there are only a few *relevant* articles in the dataset. Recall is the ratio between the number of *relevant* articles that were labeled correctly and the number of *relevant* articles that were classified. Precision is the ratio between the number of *relevant* articles that were labeled correctly by the classifier and

the number of articles that were labeled as *relevant* by the classifier. Recall and precision must always be considered together because it is possible to increase one at the cost of the other. A recall of 1 can be achieved by simply classifying all the articles as *relevant*, but precision will be low. Precision equal to 1 can be attained by classifying one *relevant* article correctly and classifying all other articles as *irrelevant*, but recall will be close to 0.

Recall and precision can be combined into one performance metric using a weighted harmonic mean, which is known as the *F-measure* [52]. It is defined as:

$$F = \frac{2 * precision * recall}{precision + recall} \quad .$$

For this work, we report recall and precision separately instead of combining them into an F-measure. Recall and precision provide information on the types of errors being made by the classifier that cannot be conveyed when these measures are combined. Furthermore, the CHAR curators stipulate that our classifier needs to demonstrate a recall greater than 0.7 and precision greater than 0.8 in order to be useful. Reporting our findings in terms of recall and precision allows us to compare our results directly to these specifications.

2.4 Related Work

In recent years, much research on automated text categorization has been conducted in the biomedical domain, as researchers and database curators are unable to keep up with the number of scientific articles published due to the ever increasing pace of biomedical

research [59, 7]. Methods have been developed to automatically detect and extract various types of biomedical information from text, including extraction of kinetic constants for protein reactions [56], identification of evidence of protein-protein interactions [11] and prediction of protein sub-cellular localizations [3]. Throughout the rest of this section we describe text classification research and competition results related to our task. We focus on work that involves determining if a document is relevant to a topic, as this is the goal of our own research.

Leonard *et al.* [33] devised a system capable of determining if a scientific article that mentions a gene or protein is, in fact, a relevant reference for the gene or protein. Their hypothesis was that the title and abstract text of articles that discuss the same gene or protein contains similar words, and those words could be used to identify additional relevant articles. The authors used articles that included a gene accession number to a database, such as GenBank [43], as positive examples of articles that were indeed relevant to the gene, and collected statistics on the words used in the title and abstract. By applying a *Bayesian estimated probability* method, the authors calculated the likelihood that a new article which mentioned a gene was a relevant reference for that gene. The authors experimented with a relevance cutoff that was adjustable in order to trade precision for recall, and vice versa. The authors reported 0.63 recall and 0.88 precision over two small, manually reviewed test sets of 148 and 150 examples.

As noted above, Schmeier *et al.* [56] proposed a method to automatically extract kinetic parameters used for biological system modeling from scientific articles.

Donaldson *et al.* [11] reported an information extraction system capable of finding potential protein-protein interactions in scientific articles. Despite the fact that the information extraction techniques used and the final goals of Schmeier and Donaldson's projects differed, they both performed the same initial step of classifying potential articles based on whether they contained the relevant information. This initial classification is necessary so that the information extraction process is performed only on relevant articles, and therefore collects relevant information.

Schmeier *et al.* used a feature vector representation based on term frequency normalized by document frequency (*tf•idf*, as discussed in Section 2.3.1). They generated terms from the full text of each document, and selected features using the *t-statistic*; the names for kinetic parameters were found to have the highest *t-value*. An SVM was trained using 400 manually labeled documents, tested on a set of 200 documents and then validated on a third set of 197 documents. The performance of the classifier on the validation set was low, with 0.5 precision and 0.31 recall.

Donaldson *et al.* used a binary feature vector to represent the text of their document abstracts, and selected features using the information gain. The authors trained both an SVM and a naïve Bayes classifier using a collection of 1,094 labeled abstracts. Both classifiers performed well in cross-validation studies; an F-measure of 0.92 was reported for the SVM, and 0.87 for the naïve Bayes classifier.

The difference between the two papers' results may be due to their dataset construction. Schmeier *et al.* used a simple keyword search to collect their documents

and used the presence of kinetic information anywhere in the text, including tables or figures, as their criterion of relevance. In contrast, Donaldson *et al.* provided very clear and specific instructions to the curators who manually constructed the datasets, and selected abstracts from within well defined topics. It is possible that Schmeier's poor classification results were caused by their unspecific approach to dataset construction; their dataset covers many different topics and, as a result, there may not be a clear difference between the *relevant* and *irrelevant* articles. However, in Donaldson's specific datasets there was a clear difference between the *relevant* and *irrelevant* abstracts, which allowed their classifiers to perform well.

The ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) holds a data mining competition every year, known as the KDD Challenge Cup. In 2002, one of the challenge tasks was to design a text classification system to aid FlyBase database curators [70]. FlyBase is a database of *Drosophila* (fruit fly) genes [14]. FlyBase curators find and read journal articles that contain gene expression information in order to add the information to the database. The challenge task had three subtasks: extracting a list of genes that were mentioned in the paper, determining automatically if a given journal article should be curated (that is, if it contained information relevant to the database), and determining if expression information was present for each of the listed genes. Performance in the curation decision and gene expression information tasks were scored using the F-measure.

The best performing system was submitted by researchers from ClearForest and Celera [51]. Their system used rule-based extraction to locate information rich areas such as the title, abstract, figure captions and footnote text and search for patterns involving gene names. Evidence of expression information was collected from the various locations and scored, and then a final decision was made on whether to recommend the paper for curation or not. The authors reported achieving an F-measure score of 0.78 on the curation decision task, which was significantly higher than the 0.62-0.64 of systems that used text categorization approaches. By carefully leveraging informative characteristics of the document, such as results for specific experiments being reported in figure captions or footnotes that included a sequence database accession number, the authors were able to create a system of rules that performed better than standard classification approaches and the systems of the other participants in the KDD Challenge Cup.

TREC, the Text REtrieval Conference [63], proposes each year problems in different categories, known as tracks, and collects and evaluates runs from the participants. The TREC Genomics track ran for 5 years, from 2003 to 2007, and involved a number of different biomedical text challenges. The 2003 task was to retrieve documents that focused on the basic biology of a target gene or its protein product from a designated organism, and the 2006 and 2007 tasks involved returning text fragments relevant to specific biomedical questions [21, 24, 25]. We discuss here the 2004 and 2005 TREC Genomic tracks, as they are the most relevant to our task.

The TREC 2004 Genomics track [22] included a task that consisted of classifying journal articles based on their relevance to the Mouse Genome Informatics (MGI) system [42]. One of MGI's goals is providing structured, coded annotation of mouse gene function by curating relevant journal articles. Automatically classifying articles based on their relevance to MGI has the potential to aid MGI curators by reducing the number of irrelevant articles they need to read and discard. TREC provided participants with a training set of 375 *relevant* and 5462 *irrelevant* articles from 2002 that had been curated by MGI, and used 420 *relevant* and 5623 *irrelevant* articles from 2003 as the test set.

All TREC 2004 Genomics Track participants reported poor results in this task, achieving F-measure scores no higher than 0.28 [22]. Cohen *et al.* tried several different approaches, selecting features using the χ^2-*statistic* and training three different classifiers, including a naïve Bayes classifier and an SVM [7]. Performance was low for both methods. Cohen attributed the classification difficulty to *conceptual drift,* that is, the process by which the language used in scientific writing changes over time. They showed that the terms that effectively differentiated between the *relevant* and *irrelevant* training documents from 2002 were very different from those that differentiate between the classes in the 2003 documents, which were used to evaluate classifier performance. We examine the effect of conceptual drift in our largest dataset in Section 3.3.

TREC 2005 included an interesting text retrieval task [23]. Text retrieval is the process of identifying documents that are relevant to a query. This is not a supervised learning task, as training examples are not provided. The TREC 2005 organizers devised

five ‘Generic Topic Types’ (GTTs); template phrases that could be filled in with specific variables (*e.g.* “Find articles describing the role of a *gene* involved in a given *disease*” where *gene* could be DRD4 and *disease* could be Alcoholism). Ten variations of each of the five GTTs (50 topics total) were provided to participants, who were then asked to locate and rank relevant articles in a 4.5 million document collection of PubMed abstracts. Results were evaluated using *mean average precision* – a metric that emphasizes ranking relevant documents high in the returned list.

The best performing system was developed by a team from York University [26]. They used external biomedical resources to find synonyms and expand the topic terms, and employed the *Okapi Basic Search System*. Okapi is based on a probability model that was developed by Robertson and Sparck Jones [53]. The Okapi BSS works by using a term weighting scheme that incorporates word, query and document frequency, along with a few tuning parameters, to probabilistically rank matching documents. They reported a mean average precision of 0.3 over the TREC 2005 topics.

In this chapter we described genome annotation and protein characterization, and introduced the CHAR database as the target application of work. We also discussed the typical approaches to text classification and surveyed literature that describes tasks that are similar to our own. Research in text classification within the biomedical domain has applied many different methods to several tasks. As we discussed, work has been done to identify relevant documents in order to extract kinetic parameters [56] or locate evidence

of protein-protein interactions [11]. However, to the best of our knowledge, a system has not been created to classify journal articles based on the presence of protein characterization experiments. We report in this thesis our efforts to create a system capable of classifying articles based on their relevance to the CHAR database.

Chapter 3

Dataset Construction

We built and used three datasets of *positive* and *negative* abstracts to train and test our classifiers, and a fourth dataset of only *positive* abstracts for validation. Each dataset consists of abstracts taken from journal articles available in the PubMed database [50]. *Positive* abstracts are taken from articles that we believe contain an experimental characterization of a protein (based on manual labeling by CHAR curators, or on labels assigned by public database curators), and, therefore, are *relevant* to the CHAR database. *Negative* abstracts are taken from articles that we believe do not contain any experimental characterization (based on the same criteria) and are *irrelevant* to CHAR. The ***Curated Journal*** and ***Curated Swiss-Prot*** datasets contain only a few hundred abstracts that were read and manually classified by CHAR curators. The third dataset, ***SP-GO***, is much larger – containing thousands of abstracts – and was created by collecting abstracts from articles used as references in online databases based on the labels assigned by the curators of the online databases. The validation dataset, ***CHAR***, was created from the articles that

were in CHAR at the onset of the project. The latter are, of course, *relevant* to CHAR, which is why this set only contains *positive* abstracts and is used for validation only.

3.1 The ***Curated Journal*** Dataset

The ***Curated Journal*** dataset consists of 96 *positive* and 107 *negative* abstracts. These abstracts were collected from every article in four issues of three different journals[2]. These journals are the most commonly referenced journals in CHAR, and three of the four issues selected contain at least one article that is referenced in CHAR. The articles were classified as being either *relevant* or *irrelevant* to CHAR, by CHAR curator and developer Dr. Madupu [38], based on their title and abstract. If the relevance could not be determined based on the abstract alone, the article was assigned to a third category: *potentially relevant*. Of the articles curated, 86.8% were classifiable as either *relevant* or *irrelevant* based on their title and abstract alone. Table 3.1 shows the distribution of curation labels assigned to the articles from each journal.

Table 3.1: Curation labels for the ***Curated Journal*** dataset. Values shown in boldface indicate the total number of articles used as *positive* and as *negative* examples in the ***Curated Journal*** dataset

	Relevant	*Irrelevant*	*Potentially relevant*	Total
J Bacteriol	71	40	20	131
J Biol Chem	13	65	6	84
Mol Microbiol	12	2	5	19
Total	**96**	**107**	31	234

[2] *Journal of Bacteriology* vol. 189, no. 5, 2007, *Journal of Bacteriology* vol. 189, no. 15, 2007, *Journal of Biological Chemistry* vol. 257, no. 19, 1982 and *Molecular Microbiology* vol. 33, no. 2, 1999.

3.2 The ***Curated Swiss-Prot*** Dataset

The ***Curated Swiss-Prot*** dataset consists of 324 *positive* and 174 *negative* abstracts. We collected the abstracts from 300 articles selected at random from entries in Swiss-Prot [61] and from 300 articles referenced by CHAR entries labeled '*dbparse*'. CHAR entries labeled '*dbparse*' were created and populated by an automatic process that used information from Swiss-Prot. The articles referenced by these '*dbparse*' entries were obtained from entries in the Swiss-Prot database. Since the CHAR curators rate the '*dbparse*' entries as unreliable (due to their generation process) we asked Dr. Madupu to label these articles along with the articles randomly selected from Swiss-Prot. Dr. Madupu read the abstracts of the 300 articles selected randomly from Swiss-Prot and the 300 articles selected from '*dbparse'* entries. She classified each article as *relevant* or *irrelevant* to CHAR, or as *potentially relevant* if she was unable to determine the relevance from the abstract alone. Of the articles that were labeled, 83% were classifiable based on their abstracts. Table 3.2 shows the breakdown of the curation labels over the two article sources.

Table 3.2: Curation labels for the ***Curated Swiss-Prot*** dataset. Values shown in boldface indicate the total number of articles used as *positive* and as *negative* examples in the ***Curated Swiss-Prot*** dataset

Swiss-Prot articles	*Relevant*	*Irrelevant*	*Potentially Relevant*	Total
At random	85	155	60	300
From '*dbparse*'	239	19	42	300
Total	**324**	**174**	102	600

3.3 The ***SP-GO*** Dataset

The ***SP-GO*** dataset, in its final form, contains 9,854 *positive* and 9,854 *negative* abstracts. The abstracts were collected from articles referenced in the well-curated public databases Swiss-Prot and GO [61, 18]. The curators for these databases include a label indicating the type of information present in the article when they incorporate it as a reference. Based on the curator's label, we collect articles that we expect to be *relevant* or *irrelevant* to the CHAR database to create a large dataset which should be as reliable as the smaller datasets that we labeled by Dr. Madupu. We describe the process of gathering the *positive* and *negative* examples, and the steps we take to correct for differences between the publication dates of the *positive* and *negative* sets, in the following paragraphs.

The *positive* abstracts in the ***SP-GO*** dataset are obtained from articles referenced by the Swiss-Prot and GO databases. We selected articles that database curators labeled as containing experimental characterization of proteins. Protein entries in the Swiss-Prot database [61] include references to PubMed articles. Each reference is associated with a '*scope*' field – a one or two word label indicating the content of the article. References with a 'CHARACTERIZATION' scope satisfy the criterion of relevance to CHAR [62]. We collected all the articles referenced by Swiss-Prot with a 'CHARACTERIZATION' scope, which provided us with 1,636 *positive* examples.

The Gene Ontology project (GO) [18] also includes references to PubMed articles. When a reference to an article is made, the GO curators assign an evidence code

that describes the type of information contained in the article. Experimental evidence codes (EXP, IDA, IPI, IMP, IGI or IEP) are assigned to articles that contain an experimental characterization [19]. These articles are supposed to be relevant to CHAR, and CHAR's developers have even constructed a new automated process to import these articles and the proteins they describe from GO into the CHAR database. We collected all articles in GO associated with experimental evidence codes, which resulted in 9,048 additional articles to use as *positive* examples.

Definite *negative* examples are more difficult to define: articles with non-experimental evidence codes in GO are not necessarily irrelevant to CHAR, while articles in Swiss-Prot that are not labeled 'CHARACTERIZATION' may still contain experimental characterization of proteins [19, 62]. Our solution is to collect abstracts from articles that are associated with proteins that are unlikely to be experimentally characterized. GenBank [43] sequence entries contain an '*experimental*' flag that is present when the sequence has been experimentally characterized. CHAR curators estimate (based on experience) that in 80-90% of the cases for which the '*experimental*' flag is not present, the protein or gene has not been experimentally characterized. We mapped 10,012 non-'*experimental*' GenBank sequences to Swiss-Prot entries using a JCVI resource called PANDA. PANDA maintains groups of accession numbers from different databases that are associated with identical protein sequences and, therefore, denote the same protein. We collected abstracts from all of the articles referenced in the Swiss-Prot entries for proteins that are synonymous with the non-'*experimental*'

GenBank sequences. This created a pool of 67,892 *negative* abstracts. Of these *negative* abstracts, 5.5% were also found in the *positive* set (which agrees with the estimate that only 80-90% of the non-'*experimental*' flagged sequences are truly uncharacterized). These overlapping abstracts were discarded from the *negative* set.

The initial ***SP-GO*** dataset held 10,684 *positive* abstracts (1,636 from Swiss-Prot and 9,048 from GO) and 10,684 *negative* abstracts selected at random from the pool of 67,892 *negative* abstracts. However, early experiments indicated that the publication dates of the articles were having an impact on the classification results. Articles in the *negative* set have lower PubMed identifiers (and therefore earlier publication dates) than those in the *positive* set (as can be seen in Figure 3.1). This skewed distribution results in older articles being classified as *negative* and newer documents being labeled by the classifier as *positive* regardless of their actual relevance to CHAR. This phenomenon, known as *conceptual drift*, where the terms used in a body of articles change over time, was also noted in biomedical text and reported on in TREC 2004 [22, 7]. Figure 3.2 shows how the classification performance (determined by 5-fold cross-validation, as described in Section 5.1) over the initial ***SP-GO*** dataset is distributed over the articles in the dataset, as a function of their PubMed IDs. Recent *positive* articles and early *negative* articles tend to be classified correctly, while early *positive* articles and recent *negative* articles are often classified incorrectly. A classifier trained using the initial ***SP-GO*** dataset thus tends to label recent articles as *positive* and early articles as *negative* regardless of their actual relevance to CHAR, which is consistent with the fact that the

positive articles in the ***SP-GO*** dataset tend to be more recent than the *negative* ones. Conceptual drift may be the cause of this phenomenon; the language used to describe the scientific research has shifted over time, and the classifier has learned that articles that use older terms are *negative*, and articles that employ newer terms are *positive*.

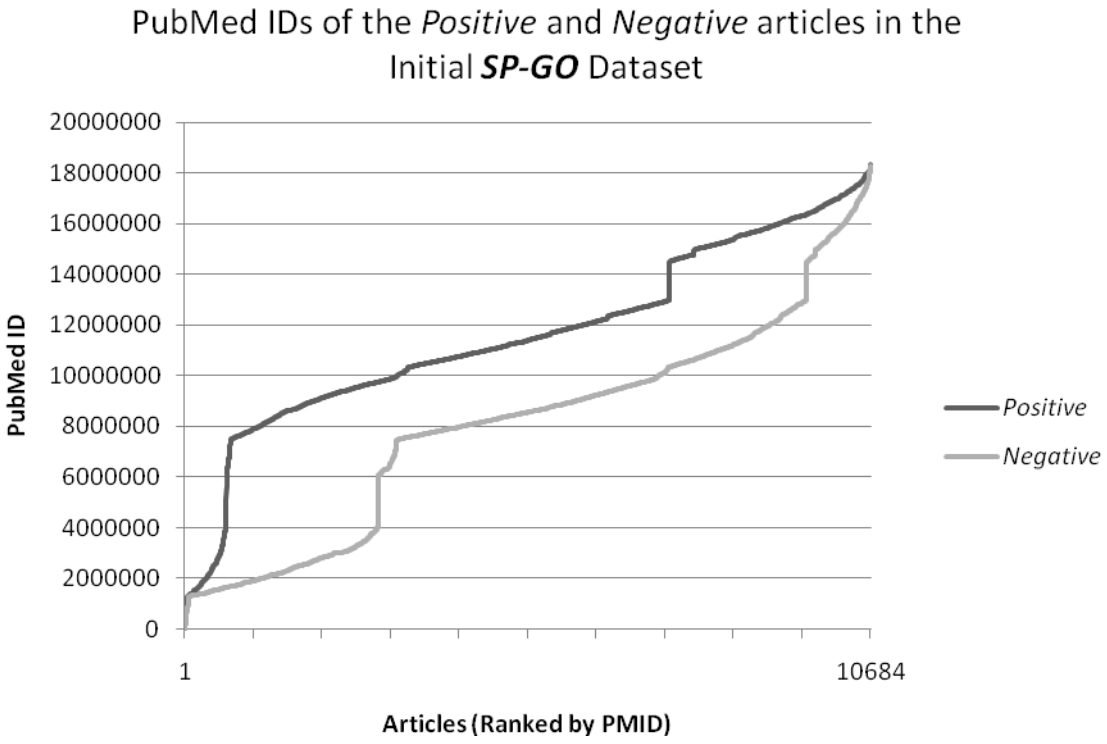
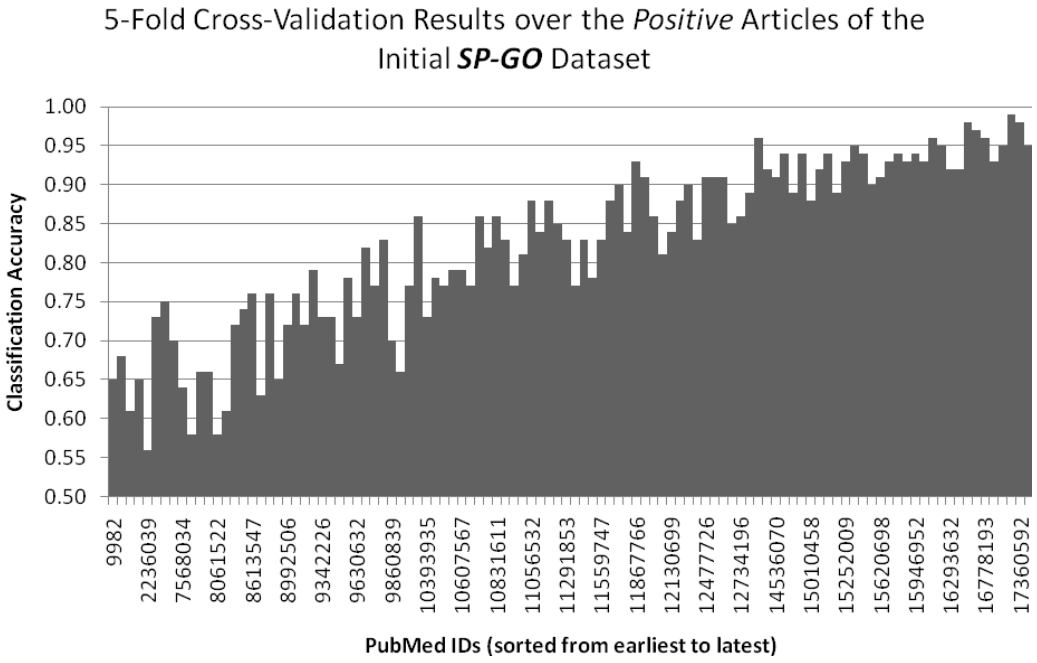


Figure 3.1: PubMed IDs for the *positive* and *negative* articles in the initial ***SP-GO*** dataset. The articles of each subset are ranked by their PMID so that the article with the lowest PMID has an ordinal value of 1, and the article with the highest PMID has an ordinal value of 10,684. The *positive* articles have a higher PubMed ID than the *negative* articles, which indicates that the *positive* articles are more recent.



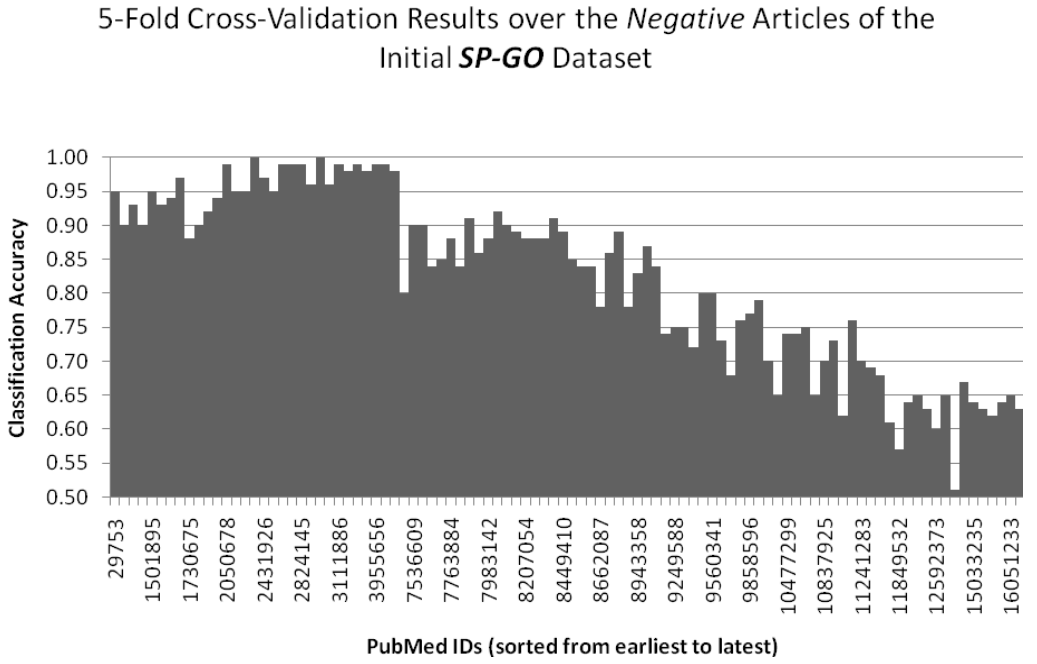


Figure 3.2: Cross-validation results over the initial ***SP-GO*** dataset, sorted in order of ascending PubMed ID. Each bar represents 100 articles. Early *positive* articles and recent *negative* articles tend to be classified incorrectly more often than the other articles.

In an effort to avoid the bias caused by conceptual drift, we took two steps to select articles for the final ***SP-GO*** dataset. First, we removed articles with PubMed ID numbers smaller (i.e. earlier) than 7600000. This removed the oldest articles from each subset, where there were many (22,080) *negative* articles but only a few (830) *positive* ones. After removing these articles, there were 1,451 *positive* abstracts from Swiss-Prot, 8,403 *positive* abstracts from GO, and the pool of *negative* abstracts was reduced to 45,812. We used all the *positive* abstracts from GO and Swiss-Prot as the *positive* portion of the ***SP-GO*** dataset, which thus consists of 9,854 abstracts. In the positive set, we noted that 6,715 abstracts are from articles with PubMed IDs smaller than 14000000 and 3,139 are from articles with PMIDs greater than 14000000. When selecting the 9,854 abstracts at random from the pool of *negative* articles, in order to obtain articles with a distribution of publication dates similar to the *positive* set, we chose 6,715 abstracts from articles with PubMed IDs smaller than 14000000 and 3,139 from articles with PMIDs greater than 14000000. Figure 3.3 shows the PubMed IDs in the new ***SP-GO*** dataset, sorted in ascending order. The distribution of the performance results (determined by 5-fold cross validation) over the articles of the new ***SP-GO*** dataset are shown in Figure 3.4. The trend of incorrectly labeling early *positive* articles and recent *negative* articles, while still present, is less pronounced than it is for the initial ***SP-GO*** trained classifier (as was shown in Figure 3.2).

The final ***SP-GO*** dataset thus consists of 9,854 *positive* abstracts collected from the Swiss-Prot and GO databases and 9,854 *negative* abstracts from the Swiss-Prot database.

3.4 The ***CHAR*** Dataset

Our fourth dataset, the ***CHAR*** dataset, contains 255 *positive* abstracts. Since the set consists only of *positive* abstracts, it is used for validation purposes only. To build it we collected abstracts from all of the articles that were referenced in manually curated CHAR entries when we began creating the datasets in June, 2008.

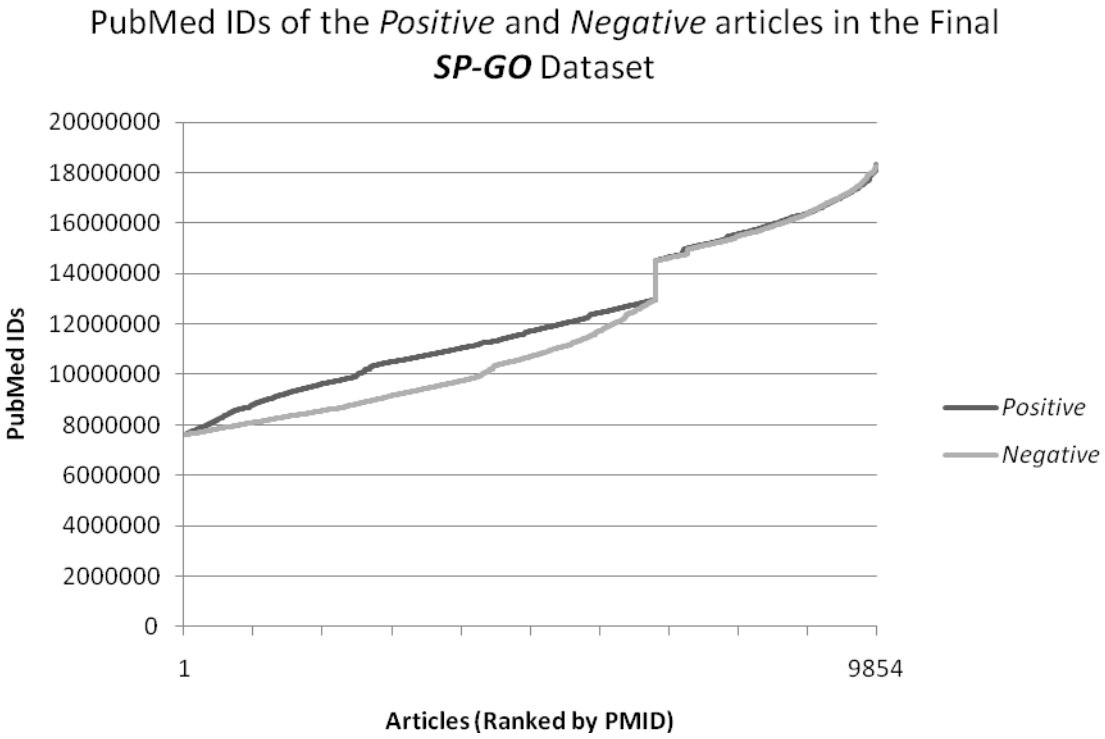


Figure 3.3: PubMed IDs for the *positive* and *negative* articles in the final ***SP-GO*** dataset. The articles of each subset are ranked by their PMID so that the article with the lowest PMID has an ordinal value of 1, and the article with the highest PMID has an ordinal value of 9,854. The *positive* and *negative* articles have much more similar PubMed IDs when compared to the initial dataset (Figure 3.1).

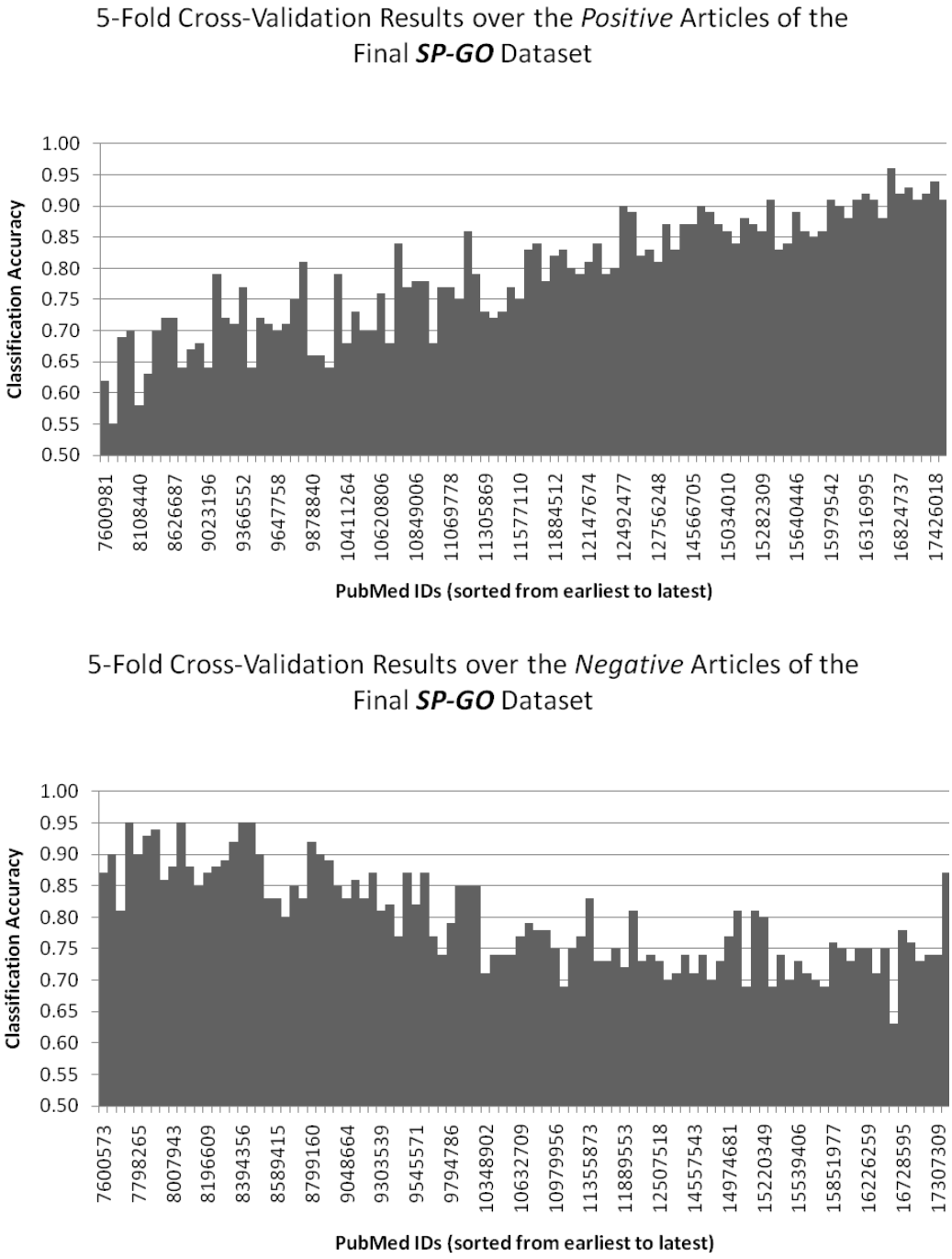


Figure 3.4: Cross-validation results over the final ***SP-GO*** dataset, sorted in order of ascending PubMed ID. Each bar represents 100 articles. Early *positive* articles and recent *negative* articles still tend to be classified incorrectly more often than the other articles, but the trend is reduced compared to Figure 3.2.

Chapter 4

Text Preparation and Classification

In Section 2.3, we introduced the steps typically taken to process and classify text. In this chapter, we describe and discuss the methods we employ for our work. We begin by explaining the process we use to convert natural language articles into machine readable vectors of features. Next we discuss our methods of feature selection; the steps we take to reduce the number of features considered by the classifier and to potentially improve classification performance. We experiment with using document frequency of a term, *Z-score* [15, 3] and removing or replacing species-specific terms, separately or in combination, to select features. Finally, we describe our classification process, including article representation, our standard and modified Naïve Bayes classifier [12, 35], and our training, testing and cross-validation process.

4.1 Text Preprocessing

As we discussed in Section 2.3.1, the text of each article needs to be processed in order to convert it into a vector of features. We collected the title and abstract for each of our articles from a copy of the PubMed database [50] which we downloaded in March, 2008.

From the abstract text, we define our terms to be single words (unigrams) and pairs of consecutive words (bigrams). In order to increase their impact on the classification, we count terms that occur in the title twice: once as part of the abstract, and once as part of the title. Title terms are valuable because they are directly related to the topic of the article. Similarly, we count terms that occur near the end of the abstract twice: once as part of the abstract, and once as a *late abstract* term. Terms near the end of the abstract are valuable because they often summarize the results or contribution of the article, and curators occasionally only need to read the last two sentences of an abstract in order to determine its relevance to CHAR [38]. We select the *n* last terms from the abstract as our late abstract terms, where *n* is an integer whose value varies in our experiments (ranging between 5 and 30).

As introduced in Section 2.3.2, stop words, such as the or and are removed, as are terms that appear to be email addresses, web addresses or numbers. The terms are stemmed using Porter's stemmer [49], which strips the suffix off words, collapsing words of different forms into their root. This allows text written in different tenses to be compared using the same features, and also reduces the number of unique terms the classifier needs to consider.

Articles are therefore represented by a collection of terms, which includes unigrams, title terms, bigrams and late abstract terms. In our experiments we include or exclude the title terms, bigrams or late abstract terms in various combinations, in order to examine their contribution to classification performance.

4.2 Feature Selection

We experimented with three methods of selecting the set of features used to represent each article. As we introduced in Section 2.3.2, feature selection has two goals: reducing the number of features, and potentially improving classification performance by removing non-informative features [69]. Without feature selection, classifiers trained on the ***SP-GO*** dataset would need to consider more than 400,000 unique terms. After feature selection, the feature set used to represent the ***SP-GO*** dataset contains between 1000 and 80,000 unique terms, depending on how we employ our feature selection methods.

Our first method of feature selection is to remove terms that are either rare or frequent based on *document frequency* thresholds. As a second method, we use the *Z-score* statistic [3, 15] to measure how effectively a feature can distinguish between classes. By discarding the terms with *Z-scores* lower than a pre-set cutoff value, we can reduce the number of features while retaining the most informative terms. For our third feature selection method, we experiment with removing or replacing *species-specific* terms, such as the scientific and common names of organisms, to avoid a bias toward certain species in our classifier. This bias is caused by the over-representation of certain species-specific terms in our *positive* datasets; a few model organisms have been studied much more thoroughly than other species, and thus have had many experimental characterization papers published about them. Additional bias is caused by the fact that the entries in the CHAR, Swiss-Prot and GO databases do not share the same distribution

with respect to the kingdoms of species (e.g. CHAR has a large proportion of prokaryotic entries when compared to Swiss-Prot or GO).

4.2.1 Feature selection using Document Frequency Thresholding

A document frequency filter is our simplest method of feature selection. We remove both rare terms and frequent terms. As we discussed in Section 2.3.2, rare and frequent terms are uninformative because they occur at almost the same rate in all articles, regardless of the class of the article, and therefore are not useful when predicting an article's class [69]. We use either the number of articles a term occurs in or the document frequency of the term (*DF*) as indicators of how frequently a term occurs. Document frequency is calculated as:

$$DF = \frac{\#\ of\ \ articles\ \ the\ term\ \ occurs\ \ in}{total\ \ \#\ of\ \ articles}. \tag{4.1}$$

When determining which terms to remove from a dataset, we consider the *positive* and the *negative* portions of the dataset separately. By considering articles from the two classes separately we are able to avoid removing predictive terms that occur very frequently in one class but only rarely in the other class. A term must fail the same condition (either fall below the minimum frequency threshold, or exceed the maximum frequency threshold) in both the *positive* and *negative* sets to be removed from the dataset. For example, if the minimum cutoff is that the term occurs in 3 articles and the maximum cutoff is that the term occurs in 60% of the articles, a term that occurs fewer than 3 times in the *positive* set of articles and fewer than 3 times in the *negative* set would

not be considered as a feature by the classifier. Likewise, a term occurring in more than 60% of the *positive* articles and in more than 60% of the *negative* articles would be removed. We note that terms whose frequency falls within the thresholds in either one of the sets (*positive* or *negative*) are not removed. To continue the previous example, if a term occurs fewer than 3 times in the *positive* articles and in more than 60% of the *negative* articles it is *not* removed from the features set used in the classification. While this term's document frequency falls outside both thresholds, namely, under the minimum in one class and above the maximum in the other, it is actually very predictive of the classes, exactly because it is a rare term in one class while frequent in the other. Such a term is therefore retained.

4.2.2 Feature selection using *Z-score*

We use the *Z-test between two proportions* [15, 66] to calculate a *Z-score* for the statistical significance of the difference between the probability of a term to occur in one set of articles and the probability of the term to occur in another set of articles. Terms whose probability to occur is statistically-significantly different between the two classes (for example, high in one class and low in the other class) are useful for predicting the class and therefore valuable for our classifier. By calculating the *Z-score* for each feature, we are able to rank features by their ability to differentiate between the *positive* and the *negative* classes. Features with a *Z-score* higher than a pre-set cutoff can then be selected for consideration by the classifier.

Let *t* denote a term, *d* denote an article and *c* denote a class: *positive* or *negative* in our case. A term, *t*, that occurs in an article, d, is denoted as $t \in d$. We denote an article, *d*, that belongs to a class, *c*, as $d \in c$. The *Z-score* measures the statistical significance of the difference between the probability of a term to be associated with one class, c_1, $\Pr(t \in d \mid d \in c_1)$, and the probability of a term to be associated with a second class, c_2, $\Pr(t \in d \mid d \in c_2)$. To estimate the conditional probability, $\Pr(t \in d \mid d \in c)$, (or $\Pr(t \mid c)$ for short) we use a maximum likelihood estimate based on the document frequency of the term in the articles from class *c*. We divide the number of articles in the class *c* that contain the term *t*, by the total number of articles in class *c* (denoted by $|D_c|$), as shown in Equation 4.2.

(4.2) $$\Pr(t \mid c) \approx \frac{\#\ of\ articles\ where\ t \in d\ and\ d \in c}{|D_c|} \quad .$$

Using the above definitions, the *Z-score* is defined as:

(4.3) $$Z^t_{c_1,c_2} = \frac{\Pr(t \mid c_1) - \Pr(t \mid c_2)}{\sqrt{\overline{P} \cdot (1-\overline{P}) \cdot \left(\frac{1}{|D_{c_1}|} + \frac{1}{|D_{c_2}|} \right)}}, where\ \overline{P} = \frac{|D_{c_1}| \cdot \Pr(t \mid c_1) + |D_{c_2}| \cdot \Pr(t \mid c_2)}{|D_{c_1}| + |D_{c_2}|} \quad .$$

The absolute score value, $|Z^t_{c_1,c_2}|$, represents the statistical significance of the difference between the estimated conditional probability for term *t* to occur in an abstract from class c_1, $\Pr(t \mid c_1)$, and the estimated conditional probability for term *t* to occur in an abstract

from class c_2, $\Pr(t \mid c_2)$. The larger the absolute *Z-score* is, the more statistically significant the difference between the probabilities is.

It is interesting to note that the *Z-score* measure is based on the magnitude of the difference between the two probabilities, and not on the ratio between the two probabilities. For example, consider two terms, *a* and *b*, and two classes, c_1 and c_2. Term *a* occurs with a probability of 0.10 in class c_1 and with a probability of 0.05 in class c_2. Term *b* occurs with a probability of 0.55 in class c_1 and with a probability of 0.45 in class c_2. Term *a* is twice as likely to appear in class c_1 than in class c_2, while term *b* occurs at a relatively similar rate in either class. Therefore, one might argue that the difference in probabilities that term *a* exhibits, between the two classes, is much more significant than that of term *b*. However, if we calculate the *Z-scores*, where the number of articles for both class c_1 and class c_2 is 100, the *Z-score* for term *a* is 1.342, while the *Z-score* for term *b* is 1.414. Thus, according to the *Z-score*, term *b* is a better predictor for the class label.

4.2.3 Replacing or removing Species-Specific Terms

Species-specific terms are terms that stem from either the scientific name, common name or strain identifier for a species (for example, *Saccharomyces cerevisiae*, budding yeast, or strain YJM789). These terms appear to be powerful predictors of class labels because they are unevenly distributed in our datasets. However, the distribution of species-specific terms in our datasets is an artifact which is present for two different reasons. The first reason is that scientific research tends to be focused on certain species, which results

in overrepresentation of these species in *positive* articles. A small number of species (such as *E. coli*, *M. musculus*, or *S. cerevisiae*) are used as model organisms and have been studied extensively by the scientific community. This results in an abundance of experimental characterization articles containing certain species-specific terms.

The second reason that some species-related terms appear at different rates in our various datasets is the overrepresentation of some kingdoms of species exhibited by the databases from which we obtained our articles. Recall that the *positive* articles in the ***SP-GO*** dataset were mostly obtained from the GO database (roughly 85% from GO and 15% from Swiss-Prot) while the *negative* articles were all obtained from the Swiss-Prot database. Among the PubMed references in the GO database (from the May 11th, 2008 release) 67% are referenced from GO entries concerned with prokaryotes, 14% with eukaryotes and 20% with viruses. In contrast, the sequence entries in Swiss-Prot were 57% prokaryote, 36% eukaryote, 4% archaea and 3% virus when we collected our articles [62]. This disparity results in an overrepresentation of prokaryote and virus species-specific terms in the *positive* articles and an overrepresentation of eukaryote species-specific terms in the *negative* articles.

Using species-specific terms as features for classification could create a bias for certain species. A classifier that uses the presence of well studied species as features indicative of relevance to CHAR may erroneously label articles about other species as *irrelevant*. Likewise, the uneven representation of species from different kingdoms in the *positive* and *negative* portions of the ***SP-GO*** dataset could result in a classifier that labels

articles based on the species they discuss rather than on the presence of protein characterization. We experiment with two methods to avoid this bias; simply removing all species-specific terms, or replacing each such term with a special *species-placeholder* term. The species-placeholder term allows the presence of any species-specific term to contribute towards classification without the bias from the overrepresentation of the specific species being mentioned.

The first step toward removing or replacing the species-specific terms in our datasets is to identify them. To this end, we downloaded a copy of NCBI's Taxonomy database [45]. This database contains scientific names, common names and synonyms for every species referenced by NCBI. We processed the names into a list of 310,088 unique terms. However, we were unable to directly use terms from the name field to identify species terms, as the entries in this field contained many instances of terms that were not exclusively species terms. The names contained such terms as *protein*, *synthetic*, *expression* or *construct* as can be seen in the following excerpt from the NCBI Taxonomy database download (Figure 4.1).

To ensure that the terms we removed from our datasets were, in fact, species-specific terms, we manually reviewed each term in our dataset that was identical to a term in the NCBI Taxonomy database. Of the 310,088 unique terms in the NCBI Taxonomy database, 5,575 terms also occur in our datasets. We confirmed that 1,793 of these terms were in fact species-specific terms. In our experiments we either remove these 1,793

terms or replace them with a common placeholder in order to eliminate species bias in our classifier.

```
TaxonID|   Name   |Name Type
111783|   Reinhardtius   |   scientific name
111784|   Greenland flounder   |  genbank common name
111784|   Greenland halibut   |   common name
111784|   Reinhardtius hippoglossoides   |  scientific name
111784|   Reinhardtius hippoglossoides (Walbaum, 1792)   |  synonym
111785|   Martes caurina   |   scientific name
111786|   eukaryotic synthetic constructs   |  scientific name
111787|   TGEV spike protein expression construct   |  scientific name
111789|   eukaryotic synthetic construct   |  scientific name
111790|   Cleistes bifaria   |  scientific name
111790|   Cleistes bifaria (Fernald) Catling & Gregg   |  synonym
```

Figure 4.1: A sample of species name entries from NCBI's Taxonomy Database. The first column contains the Taxon Identification number for the species, the second column holds the species name and the third column describes the type of name. Entries shown in boldface contain terms that are not species-specific.

A drawback of using a pre-set list to identify the species-specific terms in each article is that we only reviewed the terms that occur in our datasets. A new article may contain species-specific terms that are not in the list. Such terms will not be replaced by the species-placeholder, and therefore the presence of novel species terms in the new article will not contribute towards its classification. A more extensive list of species-specific terms would need to be developed if we intend to replace the species terms in new articles during classification.

4.3 Article Representation

We represent each article as a feature-vector, denoted as $\vec{d}$, using a *multi-variate Bernoulli* event model [35, 41]. We discussed the multi-variate Bernoulli model, along

with other article representations, in Section 2.3.1. As described in Section 4.1, the title and abstract text of each article is converted into unigrams and bigrams to create a set of terms, which is denoted as *T*. Each unique term is denoted as t_j, such that the set of all terms is $T = \{t_1,...,t_j,...,t_{|T|}\}$, where $|T|$ is the total number of unique terms. We associate a binary feature, w_j, with each of the $|T|$ unique terms. If the j^{th} term is present in the article, w_j is set to 1; otherwise w_j obtains the value of 0. Each article is therefore represented as a $|T|$-dimensional vector of binary features, denoted as: $\vec{d} = \langle w_1,..., w_j,..., w_{|T|} \rangle$.

We employ the multi-variate Bernoulli event model because we are only using the title and abstract text from each article. In applications that use full-text articles, *multinomial* models that incorporate term frequencies (as discussed in Section 2.3.1) can outperform the multi-variate Bernoulli model [41, 31]. However, for this application, we do not expect including term frequencies to improve performance because terms are unlikely to occur more than once or twice in the title and in the abstract.

4.4 Classification

We predict the relevance of articles to the CHAR database by applying a naïve Bayes classifier to the binary feature-vector that represents the article. We employ a standard naïve Bayes classifier [12, 35], training and testing it using the articles from our datasets. A naïve Bayes classifier works by assigning each article to the class that maximizes the probability of the class given the article. The classifier is trained by estimating the

conditional probabilities of each term given each class, based on the term's distribution within the training articles, and then the presence or absence of each term within each test article is used to classify the test articles. We also experiment with a modified naïve Bayes classifier that only considers the terms that are present (and disregards the terms that are absent) in the articles. The classifiers, as well as our training, testing and cross-validation process are described below.

4.4.1 Standard Naïve Bayes Classifier

Our classification task is to assign an article to a class based on its representation as a feature-vector. As we discussed in Section 4.3, the feature-vector, denoted by $\vec{d}$, is a $|T|$-dimensional vector of features, where the j^{th} feature is denoted by w_j and $|T|$ is the total number of features, such that $\vec{d} = \langle w_1, ..., w_j, ..., w_{|T|} \rangle$. The k^{th} class, denoted c_k, belongs to the set of all possible classes, C, such that $C = \{c_1, ..., c_k, ..., c_{|C|}\}$, where $|C|$ is the total number of classes. We calculate the conditional probability of each class given the article, or $\Pr(c_k \mid \vec{d})$, so that we can assign the article to the class that maximizes the probability. Using *Bayes theorem*, we can calculate the posterior probability, $\Pr(c_k \mid \vec{d})$, by multiplying the prior probability, $\Pr(c_k)$, and the likelihood, $\Pr(\vec{d} \mid c_k)$, as follows:

$$\Pr(c_k \mid \vec{d}) = \frac{\Pr(c_k) * \Pr(\vec{d} \mid c_k)}{\Pr(\vec{d})} \quad . \tag{4.4}$$

The denominator, $\Pr(\vec{d})$, is the probability of the article to have the specific combination of terms. Since we are comparing our estimates of $\Pr(c_k \mid \vec{d})$ for each $c_k \in C$, and only need to know which class is most likely given the article, that is, which class, c_k, maximizes $\Pr(c_k \mid \vec{d})$, we do not need to calculate $\Pr(\vec{d})$, because it is constant given the article $\vec{d}$. The calculation to determine the class that maximizes $\Pr(c_k \mid \vec{d})$ is thus:

$$\arg\max_{c_k \in C} \Pr(c_k \mid \vec{d}) = \arg\max_{c_k \in C} \Pr(c_k) * \Pr(\vec{d} \mid c_k) \quad . \tag{4.5}$$

The prior probability, $\Pr(c_k)$, is the probability that any article belongs to class c_k. We estimate this probability based on the number of articles in the training dataset that belong to the class.

To estimate the likelihood, $\Pr(\vec{d} \mid c_k)$, the naïve Bayes classifier uses the assumption that the probability of each term, t_j, to occur in an article is conditionally independent of the occurrence of the other terms, given the class. Under this assumption, the likelihood, $\Pr(\vec{d} \mid c_k)$, can be calculated as the product of the probabilities for each term to occur given the class:

$$\Pr(\vec{d} \mid c_k) = \prod_{j=1}^{|T|} \Pr(t_j \mid c_k) \quad . \tag{4.6}$$

Thus, Equation 4.5 becomes:

$$\arg\max_{c_k \in C} \Pr(c_k \mid \vec{d}) = \arg\max_{c_k \in C} \Pr(c_k) * \prod_{j=1}^{|T|} \Pr(t_j \mid c_k) \quad . \tag{4.7}$$

The value of the probability $\Pr(t_j \mid c_k)$ for each term t_j and class c_k is estimated from the training dataset using a maximum likelihood approach: the number of articles in class c_k that contain term t_j is divided by the total number of articles in class c_k, as shown in Equation 4.2. A *pseudo-count* is added to the number of articles that contain each term in order to guard against estimating a term's probability as zero. The conditional probability, $\Pr(t_j \mid c_k)$, for each term, along with the prior probability, $\Pr(c_k)$, for each class, are the parameters that the classifier learns from the training data.

Given a new article $\vec{d}$, to find the class c_k which maximizes the probability $\Pr(c_k \mid \vec{d})$, we use the values of the probabilities for individual terms, $\Pr(t_j \mid c_k)$, that we obtained from our training data. The article $\vec{d}$ can then be assigned to the class c_k that maximizes the probability $\Pr(c_k \mid \vec{d})$. Recall that when a term is present in an article the corresponding feature w_j is set to 1. If a term is absent, the corresponding feature w_j is set to 0. We use the probability of the term, t_j, to occur in an article given the class c_k, $\Pr(t_j \mid c_k)$, when the term is present, and the probability of the term t_j to not occur in an article given the class c_k, $1 - \Pr(t_j \mid c_k)$, when the term is absent.

The final calculation is:

$$\arg\max_{c_k \in C} \Pr(c_k \mid \vec{d}) =$$

(4.8)
$$\arg\max_{c_k \in C} \Pr(c_k) * \prod_{j=1}^{|T|} \left(w_j \Pr(t_j \mid c_k) + (1 - w_j)(1 - \Pr(t_j \mid c_k)) \right) \quad .$$

4.4.2 Only-present-terms Naïve Bayes Modification

Abstracts typically contain at most a few hundred terms, while the classifier can consider tens of thousands of terms, as we discussed in Section 4.2. For any given article, most of the terms evaluated will be absent. Because the absent terms greatly outnumber the present terms, they greatly contribute to the calculation of the probability $\Pr(c_k \mid \vec{d})$. A typical classification may be biased towards one class merely because the class is strongly indicated by the absence of a large number of terms.

In order to avoid this potential bias, and to focus exclusively on the terms present in the articles (rather than on the ones that are absent), we experiment with a modified version of the standard naïve Bayes classifier described in Section 4.4.1. The modified, *only-present-terms* naïve Bayes classifier only considers the terms present in the article, or $t_j \in d$, instead of using both the present and absent terms, to estimate which class, c_k, maximizes the probability, $\Pr(c_k \mid \vec{d})$. Thus, Equation 4.7 becomes:

(4.9)
$$\arg\max_{c_k \in C} \Pr(c_k \mid \vec{d}) \approx \arg\max_{c_k \in C} \Pr(c_k) * \prod_{t_j \in d} \Pr(t_j \mid c_k) \quad .$$

When we exclude the absent terms in the calculation shown in Equation 4.8, we determine the most probable class based only on the present terms. Recall that when a term, t_j, is present in the article, $\vec{d}$, its weight, w_j, is set to 1. Otherwise, the weight, w_j, is 0. Therefore, the *only-present-terms* calculation to estimate the class, c_k, that maximizes the probability $\Pr(c_k \mid \vec{d})$ can be written as:

$$\arg\max_{c_k \in C} \Pr(c_k \mid \vec{d}) \approx \arg\max_{c_k \in C} \Pr(c_k) * \prod_{j=1}^{|T|} \left(w_j \Pr(t_j \mid c_k) + (1 - w_j) \right) \quad . \tag{4.10}$$

Due to the omission of terms, we are not truly calculating the class that maximizes the probability $\Pr(c_k \mid \vec{d})$. The value we calculate is highest for the class that is most probable given the terms that are present in the article, $\vec{d}$, without considering the terms that are not part of the article. Thus, the only terms that impact the decision regarding which class the article is assigned to are the terms present in the article.

In practice, this modification appears to slightly improve classification performance in many cases. We compare the standard and only-present-terms naïve Bayes classifiers throughout our results in Sections 5.3 and 5.6. Further testing and investigation of the probabilistic ramifications of the only-present-terms naïve Bayes modification presented here needs to be carried out before it can be concluded that the observed improvement is reproducible and ubiquitous.

Chapter 5

Results

In this chapter we describe the experiments we performed to investigate how effective a naïve Bayes classifier is when trained using our datasets, and report the results of these experiments. First we outline our test methods and experimental settings, and discuss the metrics we use to evaluate performance. Next we describe the cross-validation and test performance of classifiers trained using the ***SP-GO*** dataset (introduced in Section 3.3) as well as classifiers trained using each of the two curated datasets (described in Section 3.1 and 3.2). We examine the Kullback-Leibler divergence between the *positive* and *negative* portions of our various datasets. By leveraging the Kullback-Leibler divergence results, we create a new dataset from a subset of the ***SP-GO*** dataset and report its cross-validation and test performance.

5.1 Experiment Design

We use two methods and various feature selection settings to evaluate classifier performance. First, we perform 5-fold cross-validation to train and test the classifier on the same dataset without exposing the test data to the classifier during the training

process [28]. The dataset is split at random into 5 subsets of equal size using a stratified approach: each subset contains the same proportion of *positive* and *negative* examples. A classifier is trained on 4 of the subsets, and then used to classify the articles in the left-out subset. This process is repeated 5 times in total, with each subset being used as the test set once. We repeat the 5-fold cross-validation experiment 10 times, using a different random partition of the dataset each time. The 10 times repetition allows us to evaluate whether the results are consistent across different random partitions. We report the means of the performance measures, as well as the standard deviations, as an indication of the stability of the results.

Second, we train the classifier using the whole dataset and then test the classifier on the articles in the small curated datasets (***Curated Journal*** and ***Curated Swiss-Prot***), and on the ***CHAR*** validation set. We expect that performance over the ***Curated Journal*** dataset is indicative of performance on articles from any issue of the three journals we curated, and provides insight into how the classifier may perform over articles from similar journals. Performance on the ***Swiss-Prot*** dataset is a likely predictor of how well the classifier will perform on articles referenced by the ***Swiss-Prot*** database. The performance over the ***CHAR*** dataset is shows whether the classifier is consistent with the current CHAR curation process. However, since the ***CHAR*** dataset only contains *relevant* articles, high performance over the ***CHAR*** dataset only indicates high recall of *relevant* articles – it says nothing about performance on *irrelevant* articles. Ideally, we

expect our best classifier to show good performance over the curated datasets as well as over the ***CHAR*** dataset.

To evaluate classification performance we report our results in terms of the commonly used metrics recall, precision and accuracy, which we discussed in Section 2.3.4. According to CHAR's curators, for a classifier to be useful, it needs to demonstrate recall greater than 0.7 and precision greater than 0.8 [38]. Recall is the ratio between the number of articles correctly labeled as *positive* by the classifier and the number of truly *positive* articles. Precision is the ratio between the number of articles correctly labeled as *positive* by the classifier and the number of articles that were labeled *positive* by the classifier, both correctly and incorrectly. Accuracy is the ratio between the number of articles, *positive* or *negative*, that were labeled correctly by the classifier and the number of articles the classifier considered. Each metric has a minimum value of 0 and a maximum value of 1.

5.1.1 Feature Selection

We repeat the 10 times 5-fold cross-validation and train/test evaluation of each dataset several times, varying the set of features used to represent the articles. There are four types of feature selection we experiment with: filtering by *document frequency*, filtering by *Z-score*, replacing or removing *species-specific* terms, and including or excluding title terms, bigrams or *late abstract* terms.

When we employ the document frequency cutoffs (as described in Section 4.2.1) we remove the terms that are either more rare than a minimum cutoff value, or more

frequent than a maximum cutoff value. The minimum cutoff, which is the minimum number of articles a term must occur in to be included in the feature set, is varied between 3 articles and 100 articles. The maximum cutoff, which is the maximum number of articles a term may occur in to be included in the feature set, is varied between 10% of the articles and 60% of articles in the dataset.

The *Z-score* filtering (Section 4.2.2) ranks terms by *Z-score* and removes terms whose score falls below a predefined threshold. During our experiments, we vary the *Z-score* threshold between 0 and 5. When the threshold is 0 no terms are removed from consideration by the classifier. When the threshold is greater than 5, very few terms (fewer than 1000 for the ***SP-GO*** dataset, and fewer than 100 for the other datasets) are involved in the classification and performance decreases.

We employ two methods for handling species-specific terms (Section 4.2.3). In our experiments, we either remove all species-specific terms or replace all these terms with a special *species-placeholder* term. In order to also examine the effect of not removing or replacing species-specific terms, for some of our experiments we do not employ either of these methods and leave the species-specific terms intact.

When articles are converted into feature-vectors (Section 4.1) we use unigrams and bigrams from the title and abstract as terms. We count terms from the title and from the end of the abstract twice, in order to increase their impact on the classification. During our experiments, we include or exclude bigram terms, title terms and late abstract terms in different combinations, in order to evaluate their contribution to the

classification performance. We also vary our definition of a late abstract term by changing the number of terms from the end of each abstract we count twice between 5 and 30. We do not experiment with removing unigrams.

5.1.2 Standard and Only-Present-Terms Naïve Bayes Classifiers

We investigate two different calculation methods to determine the most likely class for a given article. First is the standard naïve Bayes calculation (defined in Section 4.4.1), which uses both the terms that are present and the terms that are absent to assign an article to a class. Second is our modified, *only-present-terms* calculation (defined in Section 4.4.2), which, as the name suggests, only uses the present terms and disregards the absent ones. We experiment with each calculation approach during the evaluation of our classifiers.

5.1.3 Feature Set Optimization

When evaluating each classifier, we search for the set of features that maximizes classifier performance. We repeat the 10 times 5-fold cross-validation and train/test experiments using different parameters for the feature selection methods (as described in Section 5.1.1). A grid search, where we iterate over and evaluate every combination of parameters for the four feature selection methods discussed above, is a technique to exhaustively discover the optimal set of features. However, the computation cost of this approach is prohibitively high, as it would result in $m*n*p*q$ ten times 5-fold cross

validation experiments and train/test evaluations[3]. Instead we examine filtering by document frequency, filtering by *Z-score*, removing or replacing species-specific terms, and including or excluding bigram, title and late abstract terms separately. When we experiment with one feature selection method, we use the default parameters for the other methods, at a computation cost of $m+n+p+q$ evaluations. While we are unlikely to determine the optimal feature set using this approach, examining the feature selection methods one at a time is much more feasible and allows us to draw conclusions about each individual feature selection method.

When we examine the effect one feature selection method has on the classifier's performance, we set the parameters for the other feature selection methods to their default values, which are shown in Table 5.1. When we employ each of the feature selection methods using their default parameters, we refer to the resulting feature set as the *default feature set*. We report the results from classifiers trained using the default feature set as well as an overview of the feature set optimization search for each dataset in this chapter. A complete listing of performance results and term counts over all feature sets can be found in Appendices B, C and D.

[3] Where m is the number of document frequency cutoffs examined, n is the number of *Z-score* cutoffs examined, p is the number of species-specific term approaches examined and q is the number of combinations of bigrams, title and late abstract terms examined.

Table 5.1: Default Feature Selection Parameters

Feature Selection Method	**Cutoff/Terms included or excluded**
Minimum document frequency cutoff	Terms that occur in fewer than 3 articles are excluded
Maximum document frequency cutoff	Terms that occur in more than 60% of the articles are excluded
Z-Score cutoff	No terms are excluded based on *Z-score*
Species-specific terms	Species-specific terms are included
Term set (unigrams are always included)	Title and bigram terms are included, late abstract terms are excluded

5.2 Results for Classifiers trained on the ***SP-GO*** Dataset

The ***SP-GO*** dataset, as described in Section 3.3, is our largest dataset, which we initially intended to use as our training dataset. It consists of 9,854 *positive* articles selected based on curator annotations from the Swiss-Prot and GO databases, and 9,854 *negative* articles selected from Swiss-Prot entries that are unlikely to have been experimentally characterized. We trained and tested classifiers on the ***SP-GO*** dataset using 10 times 5-fold cross-validation. We also trained classifiers on the complete ***SP-GO*** dataset and then tested them on the other datasets. In order to evaluate performance over several different feature sets, we repeated the cross-validation and testing using classifiers trained under various conditions. Classifiers trained on the ***SP-GO*** dataset perform well in cross-validation but poorly when tested on the ***Curated Journal***, ***Curated Swiss-Prot*** and the ***CHAR*** validation datasets. Given the fact that the ***SP-GO*** dataset was built using articles annotated by public database curators as containing experimental characterization, and

the fact that we had thousands of training articles in the dataset, we expected classifiers trained on the ***SP-GO*** dataset to be an effective predictor of relevance to CHAR. However, classifiers trained on the ***SP-GO*** dataset mislabel more than half of the articles in the curated and ***CHAR*** datasets. We present an overview of the cross-validation and test results for classifiers trained on the ***SP-GO*** dataset in this section. The full classification results over all the feature sets evaluated, as well as the number of terms involved in the classification under different feature sets, are given in Appendix B.

5.2.1 Cross-Validation Results for Classifiers trained on the ***SP-GO*** Dataset

We performed multiple 10 times 5-fold cross-validation experiments over the ***SP-GO*** dataset. For each group of ten 5-fold runs, we used a different feature set. Each feature set was evaluated using both the standard calculation of probabilities within the naïve Bayes classifier and our modified only-present-terms calculation. We varied the feature set using the document frequency, *Z-score* and species-specific term feature selection methods, and by including or excluding title, bigram and late abstract terms as shown in Table 5.2. Each of the four feature selection methods was examined separately. When we varied the parameters of one feature selection method, we set the parameters of the other methods to their default values (given in Table 5.1). We present an overview of the results here. A complete listing of the cross-validation results is given in Appendix B.1.

When using the standard naïve Bayes calculation, the 10 times 5-fold cross-validation performance of classifiers trained using the ***SP-GO*** dataset is very consistent. The recall, precision and accuracy over all the evaluated feature sets are between 0.72

and 0.74 with a very low (less than 0.002) standard deviation. Cross-validation results for a classifier built using the default feature set are shown in Table 5.3.

Table 5.2: Feature Selection Parameters evaluated with Classifiers trained on the ***SP-GO*** Dataset

Feature Selection Method	**Cutoffs/Terms included or excluded**
Minimum document frequency cutoff	Terms occurring in less than 3, 10, 30 or 100 articles are excluded
Maximum document frequency cutoff	Terms occurring in more than10%, 20%, 30%, 40% or 60%[4] of the articles are excluded
Z-Score cutoff	0 (no *Z-score* filtering), 2, 3, 4, 5
Species-specific terms	Species-specific terms included, Species-specific terms removed, Species-specific terms replaced with placeholder
Term Set (unigrams are always included)	Only unigrams, Title terms included, Bigram terms included, Title and bigram terms included, Last 10 abstract terms included, Last 20 abstract terms included, Last 30 abstract terms included, Title, bigram and last 10 abstract terms included, Title, bigram and last 20 abstract terms included, Title, bigram and last 30 abstract terms included

Table 5.3: Cross-validation results for the standard naïve Bayes classifier trained on the ***SP-GO*** dataset over the default feature set. Mean and standard deviation for each measure, calculated over the ten 5-fold runs, are shown

	Mean	Standard Deviation
Recall	0.7333	0.00117
Precision	0.7343	0.00138
Accuracy	0.7343	0.00115

[4] For the ***SP-GO*** dataset, the same terms are involved in classification at both the 50% and the 60% maximum document frequency cutoffs, therefore we only include the 60% results.

When using the modified, only-present-terms naïve Bayes calculation, the performance tends to be slightly better than it is for the standard naïve Bayes calculation, as can be seen in Appendix B.1. The mean recall, precision and accuracy from each cross-validation experiment are very rarely below 0.72 and are occasionally higher than 0.74. The standard deviation around the mean for each measure from all the cross-validation experiments was lower than 0.002. For many of the feature sets we evaluated, the recall appeared slightly higher than the precision, which is unlike how these two measures tended to be nearly equal when using the standard naïve Bayes classifier. Higher recall indicates that the only-present-terms naïve Bayes classifier tends to label articles as *positive* more often than the standard naïve Bayes classifier. Table 5.4 lists the cross-validation results for training a classifier using the ***SP-GO*** dataset and the default feature set, where the calculations only use present terms.

Table 5.4: Cross-validation results for the only-present-terms naïve Bayes classifier trained on the ***SP-GO*** dataset over the default feature set. The means and standard deviations shown, are calculated over of the ten 5-fold runs

	Mean	Standard Deviation
Recall	0.7412	0.00133
Precision	0.7296	0.00121
Accuracy	0.7336	0.00080

5.2.2 Test Results for Classifiers trained on the ***SP-GO*** Dataset

We trained classifiers using the ***SP-GO*** dataset over various feature sets, employing both the standard naïve Bayes calculation and our modified only-present-terms calculation,

and tested them, on the ***Curated Journal***, ***Curated Swiss-Prot*** and ***CHAR*** datasets. All feature selection methods, with the exception of document frequency, were evaluated over the cutoffs and term sets shown in Table 5.2. To reduce the number of tests, we examine fewer feature sets created using the document frequency cutoffs than we did during cross-validation. The minimum cutoff was set to exclude terms that occur in fewer than 3, 30, and 100 articles. The maximum cutoff was set to exclude terms that occur in more than 10%, 30% and 60% of the articles. While evaluating one feature selection method, the other methods were held to their default parameters (as shown in Table 5.1). The full results are provided in Appendix B.2.

When training classifiers over the ***SP-GO*** dataset and testing on the curated and ***CHAR*** datasets, using the standard naïve Bayes calculation, we found performance to be very poor. The best performing classifier, over any of the feature sets evaluated, uses 100 as the minimum document frequency cutoff, 30% as the maximum document frequency cutoff. Table 5.5 shows the performance for this classifier.

Table 5.5: Test results for a classifier trained using the ***SP-GO*** dataset and tested on the other datasets, excluding terms that occur in less than 100 articles and terms that occur in more than 30% of the articles

	Curated Journal	***Curated Swiss-Prot***	***CHAR***
Recall	0.625	0.2222	
Precision	0.4511	0.6102	
Accuracy	0.4631	0.4016	0.349

As can be seen in Appendix B.2, classification performance was typically worse than that shown in Table 5.5, with accuracy on the ***Curated Journal*** dataset around 0.4, accuracy

on the ***Curated Swiss-Prot*** dataset around 0.35, and accuracy on the ***CHAR*** dataset around 0.27. Performance varied as different feature sets were evaluated but the results were consistently poor.

When trained on the ***SP-GO*** dataset, classifiers that employ the only-present-terms calculation demonstrate performance similar to that of classifiers that employ the standard naïve Bayes calculation, as can be seen in Appendix B.2. Performance was, again, highest for the classifier that did not consider terms that occurred in fewer than 100 articles or terms that occurred in more than 30% of the articles. Much like in the standard calculation results, the accuracy of all the classifiers was well below 0.5. Accuracy on the ***Curated Journal*** dataset was typically around 0.4, on the ***Curated Swiss-Prot*** dataset about 0.35, and on the ***CHAR*** validation set accuracy was around 0.28.

5.3 Results for Classifiers trained on the Curated Datasets

The ***Curated Journal*** dataset (described in Section 3.1) consists of articles selected from four issues of three journals that were manually labeled as either *relevant* or *irrelevant* to CHAR by a curator. The ***Curated Swiss-Prot*** dataset (described in Section 3.2) consists of articles collected from the Swiss-Prot database that were also manually labeled. As these datasets only contain a few hundred articles, we initially intended to use them to validate classifiers trained on the ***SP-GO*** dataset. However, given the poor performance of the ***SP-GO*** classifier, we investigated classifiers trained on the curated datasets in order to determine whether the *positive* and *negative* articles are indeed distinguishable and "classifiable". We trained and tested classifiers on each dataset using 5-fold cross-

validation (as described in Section 5.1). We also trained a classifier on each of the two curated datasets and tested it on the other curated dataset, as well as on the ***CHAR*** validation set. The classifiers were all trained using the default feature set (Table 5.1), and tested while using the standard naïve Bayes calculation to determine the most likely class for each article. The classifiers perform well in cross-validation, with results comparable to – or better than – the cross-validation results obtained by the ***SP-GO*** classifier. Furthermore, each classifier trained on one dataset performs well when tested on the other dataset and on the ***CHAR*** validation set, correctly classifying many more articles than the ***SP-GO*** classifier. We report results over the default feature set here. A full listing of the cross-validation, test and term count results are presented in Appendix C.

5.3.1 Results for Classifiers trained on the ***Curated Journal*** Dataset

Classifiers trained on the ***Curated Journal*** dataset perform well in cross-validation, with mean recall, precision and accuracy all between 0.71 and 0.73. The standard deviation of these values over the ten cross-validation runs is less than 0.03 for recall and less than 0.02 for precision and accuracy.

Test results for a classifier trained on the ***Curated Journal*** dataset and tested on the ***Curated Swiss-Prot*** and ***CHAR*** datasets are shown in Table 5.6. Classifiers trained on the ***Curated Journal*** dataset perform unexpectedly well, given that the dataset is small, containing only 96 *positive* articles and 107 *negative* articles.

Table 5.6: Test results for a classifier trained on the ***Curated Journal*** Dataset

	Curated Swiss-Prot	***CHAR***
Recall	0.8086	
Precision	0.7298	
Accuracy	0.6807	0.8037

5.3.2 Results for Classifiers trained on the ***Curated Swiss-Prot*** Dataset

Classifiers trained on the ***Curated Swiss-Prot*** dataset perform very well in cross-validation. The mean recall over the ten 5-fold runs is 0.89, mean precision is 0.84 and mean accuracy is 0.82. The standard deviation around each value was lower than 0.01.

Test results for a classifier trained on the ***Curated Swiss-Prot*** dataset and tested over the ***Curated Journal*** and ***CHAR*** datasets are shown in Table 5.7.

Table 5.7: Test Results for a classifier trained on the ***Curated Swiss-Prot*** Dataset

	Curated Journal	***CHAR***
Recall	0.9375	
Precision	0.5233	
Accuracy	0.5665	0.8778

As can be seen in Table 5.7, classifiers trained using the ***Curated Swiss-Prot*** dataset also perform unexpectedly well given the number of articles in the dataset (324 *positive* and 174 *negative* articles), although precision and accuracy are low. Other classifiers we tested over the ***Curated Journal*** dataset also exhibit low precision and accuracy (Section 5.5.2), and we further discuss this result in Section 6.2.

5.4 Kullback-Leibler Divergence Analysis of Term Distributions

In order to examine the differences between our various datasets, we calculated the Kullback-Leibler (KL) divergence [29] among the term distributions of the different datasets. KL divergence is a measure of the difference between two probability distributions. Here, the probability distribution for each dataset is a multinomial representing the respective term frequencies of all the terms used to represent the articles. The KL divergence is non-negative; two identical probability distributions have a KL divergence of 0. Higher KL divergence values indicate a greater difference between the distributions, while lower values indicate greater similarity. We use a symmetric version [29] of the Kullback-Leibler divergence, which is defined for two multinomial probability distributions, P and Q where $P = \{p_1,...,p_n\}$ and $Q = \{q_1,...,q_n\}$, as:

$$KL(P,Q) = \sum_{i=1}^{n} (p_i - q_i) \log_2 \frac{p_i}{q_i} \quad . \tag{5.1}$$

We calculate the probability for a term to occur in the articles of a dataset using the same approach as we do for the naïve Bayes classifier (Section 4.4.1), adding *pseudo-counts* so that no estimate for the probability of a term to occur is zero. To avoid using many terms that are represented only by *pseudo-counts* (as our largest datasets have many more unique terms than our small ones) we calculate the KL divergence using a distribution over a specific subset of the terms. First, we select all the terms from the ***Curated Journal*** dataset (our smallest dataset, with the least unique terms) that pass our default document frequency filter (terms must occur in more than 3 articles and in fewer

than 60% of the articles in at least one class, *positive* or *negative*, of the dataset). This results in a collection of 1,207 terms. In order to retain the terms that are most significant in the ***SP-GO*** and ***Curated SwissProt*** datasets, we add to the collection terms with a *Z-score* higher than 10 from the ***SP-GO*** dataset (189 terms). Adding terms with a *Z-score* higher than 10 strikes a balance between including the most important terms from the ***SP-GO*** dataset and not adding many terms that do not occur in the smaller datasets. We also add the top 50 terms, by *Z-score*, from the ***Curated SwissProt*** dataset, again with the goal of including the most important terms. 79 of the terms from the ***SP-GO*** and ***Curated SwissProt*** datasets were not already present in the collection of 1,207 terms from ***Curated Journal***. The final list consists of 1,286 terms. We calculate our Kullback-Leibler divergence values over the multinomial frequency distribution of these terms.

5.4.1 KL Divergence Results

We calculated the Kullback-Leibler divergence between all of our datasets, separating each dataset into two sets; the *positive* articles and the *negative* articles. Ideally, the KL divergence between two *positive* sets, or between two *negative* sets, should be low (their term distributions should be similar). The divergence between a *positive* and a *negative* set should be higher (which is indicative of much difference). Table 5.8 shows the KL divergence values between every pair of our positive and negative datasets.

Table 5.8: Kullback-Leibler Divergence between dataset pairs (***CurJol*** denotes the ***Curated Journal*** dataset, ***CurSP*** denotes the ***Curated Swiss-Prot*** dataset. *Pos* indicates the *positive* portion of the dataset, *Neg* indicates *negative*).

	CurJol *Pos*	***CurJol*** *Neg*	***CurSP*** *Pos*	***CurSP*** *Neg*	***CHAR*** (*Pos*)
SP-GO *Pos*	1.1442	1.2714	0.9622	0.5624	0.9224
SP-GO *Neg*	1.1733	1.3784	0.5790	0.2200	0.6752
CurJol *Pos*			0.8167	1.5319	0.7302
CurJol *Neg*			1.4791	1.7227	1.3580
CurSP *Pos*					0.3175
CurSP *Neg*					0.9667

According to the Kullback-Leibler divergence values seen in Table 5.8, the ***Curated Journal*** *positives*, ***Curated Swiss-Prot*** *positives* and ***CHAR*** *positives* all appear to have term distributions that are relatively similar (low divergence), when compared to the divergence between these *positives* and the corresponding *negatives* from the respective datasets. These values are consistent with the effective performance of classifiers trained on the curated datasets. However, the term distribution of the ***SP-GO*** *positive* subset is almost equally dissimilar to the term distribution of the *positive* subset as it is to the *negative* subset of the ***Curated Journal*** dataset. Moreover, the ***SP-GO*** *positive* subset appears less similar than the ***SP-GO*** *negative* set is to the *positive* articles from ***Curated Swiss-Prot*** and ***CHAR***. These values show that the ***SP-GO*** *positives* are very different from the other *positive* sets. Therefore, training on the ***SP-GO*** dataset leads to many classification errors on the articles from the other datasets, as reported in Table 5.5. The Kullback-Leibler divergence results indicate that the term distributions of

the *positive* articles from the ***SP-GO*** dataset, in particular, are responsible for the low performance.

5.4.2 Separating the ***SP-GO*** Dataset

In order to better understand the properties of the ***SP-GO*** positive articles, we separated them into two subsets based on their database of origin. The ***SPonly*** set contains the 1,451 *positive* articles from ***SP-GO*** that were collected from the Swiss-Prot database. The ***GOonly*** set contains the 8,403 *positive* articles that we collected from the GO database. Table 5.9 shows the Kullback-Leibler divergence of the term distributions in the ***SPonly*** and ***GOonly*** subsets with respect to the term distributions in the *positive* and *negative* articles from the other datasets.

Table 5.9: KL divergence between each of the ***SPonly*** and ***GOonly*** sets and all other subsets. (***CurSP*** denotes the ***Curated Swiss-Prot*** dataset; ***CurJol*** denotes the ***Curated Journal*** dataset. *Pos* indicates the *positive* portion of the dataset, *Neg* indicates *negative*).

	CurJol *Pos*	***CurJol*** *Neg*	***CurSP*** *Pos*	***CurSP*** *Neg*	***CHAR*** (*Pos*)
SPonly (*Pos*)	0.9541	1.2336	0.3825	0.5031	0.5010
GOonly (*Pos*)	1.3226	1.3920	1.2130	0.6744	1.1381

From Table 5.9, it can be seen that the term distribution of articles that Swiss-Prot curators labeled as 'CHARACTERIZATION' (***SPonly***) are more similar to (i.e. show a lower-KL divergence) the *positive* articles from the ***Curated Journal***, ***Curated Swiss-Prot*** and the ***CHAR*** datasets than to the respective *negative* articles. In contrast, articles labeled with experimental evidence codes in the GO database (***GOonly***) appear dissimilar

to the *positives* articles in the curated and the ***CHAR*** datasets, and are actually most similar to the *negative* articles from the ***Curated Swiss-Prot*** dataset.

5.5 Results for Classifiers trained on the ***SPonly*** Dataset

The ***SPonly*** dataset consists of the 1,451 *positive* ***SP-GO*** articles that we collected from the Swiss-Prot database and 1,451 of the *negative* ***SP-GO*** articles selected at random. The Kullback-Leibler divergence results shown in Table 5.8 and in Table 5.9 suggest that a classifier trained using the ***SPonly*** dataset will perform better than one trained on the ***SP-GO*** dataset discussed in Section 5.2. We investigated classifiers trained and tested on the ***SPonly*** dataset using cross-validation. We also evaluate classifiers trained on the ***SPonly*** dataset by testing over the ***Curated Journal***, ***Curated Swiss-Prot*** and ***CHAR*** datasets. Classifiers trained on the ***SPonly*** dataset perform well in cross-validation and very well when tested on the curated and the ***CHAR*** datasets. In some cases, performance surpasses the 0.7 recall and 0.8 precision requirements specified by CHAR curators. The results are discussed throughout the rest of this section. A full listing of all ***SPonly*** cross-validation runs and test results, as well as term counts, are provided in Appendix D.

5.5.1 Cross-Validation Results for Classifiers trained on the ***SPonly*** Dataset

We performed several cross-validation studies of classifiers trained on the ***SPonly*** dataset, varying the feature set and the class calculation method, as described in Section 5.1. We vary the feature selection cutoffs and the terms that are included, as shown in

Table 5.10. Note that, in addition to what was evaluated on the ***SP-GO*** dataset (Section 5.2.1), we increase the granularity of our examination of the late abstract terms by evaluating including the last 5 and last 15 terms from the abstract. When evaluating one feature selection method, we set the parameters of the other methods to their default values (Table 5.1). The full results are listed in Appendix D.1.

Table 5.10: Feature Selection Parameters evaluated with Classifiers trained on the ***SPonly*** Dataset

Feature Selection Method	**Cutoffs/Terms included or Excluded**
Minimum document frequency cutoff	Terms that occur in less than 3, 10, 30 or 100 articles are excluded
Maximum document frequency cutoff	Terms that occur in more than10%, 20%, 30%, 40%, 50% or 60% of the articles are excluded
Z-Score cutoff	0 (no *Z-score* filtering), 0.5, 1, 2, 3, 4, 5
Species-specific terms	Species-specific terms included, Species-specific terms removed, Species-specific terms replaced with placeholder
Term Set (unigrams are always included)	Only unigrams, Title terms included, Bigram terms included, Title and bigram terms included, Last 5 abstract terms included, Last 10 abstract terms included, Last 15 abstract terms included, Last 20 abstract terms included, Last 30 abstract terms included, Title, bigram and last 5 abstract terms included, Title, bigram and last 10 abstract terms included, Title, bigram and last 15 abstract terms included, Title, bigram and last 20 abstract terms included, Title, bigram and last 30 abstract terms included

Table 5.11 shows the performance of a standard naïve Bayes classifier trained with the default feature set. As can be seen in Appendix D.1, when using the standard

naïve Bayes calculation ***SPonly*** cross-validation performance is consistent across recall, precision and accuracy. The highest performance obtained for all three measures is 0.72. The lowest recall obtained was 0.62, lowest precision was 0.70 and lowest accuracy was 0.68. The standard deviation of the recall, precision and accuracy over the ten 5-fold runs was consistently less than 0.005.

Table 5.11: Cross-validation results for standard naïve Bayes classifiers trained on the ***SPonly*** dataset using the default feature set. Means and standard deviations over the ten 5-fold runs are shown

	Mean	Standard Deviation
Recall	0.7112	0.0066
Precision	0.7161	0.0045
Accuracy	0.7144	0.0036

Classifiers trained using the ***SPonly*** dataset and employing the modified, only-present-terms calculation to determine class assignments show slightly lower performance than the same classifiers using the standard naïve Bayes calculation, as shown in Appendix D.1. Over the different feature sets we used with the only-present-terms classifier trained on the ***SPonly*** dataset there is greater variance in the results when compared to our other cross-validation experiments. While the accuracy and, even more so, the precision of the only-present-terms classifiers are lower than their standard calculation counterparts, recall is much higher, approaching 0.85 in some cases, and 0.9 in one instance. Recall never falls below 0.75. The high recall along with lower precision and accuracy indicate that the only-present-term classifier tends to label articles as *positive* more often than the standard classifier, regardless of the actual class of the

articles. Precision performance tends to be between 0.6 and 0.7, and accuracy varies between 0.65 and 0.71. Performance results for an only-present-terms classifier trained with the default feature set are shown in Table 5.12.

Table 5.12: Cross-validation results for only-present-term naïve Bayes classifiers trained on the ***SPonly*** dataset using the default feature set. Means and standard deviations over the ten 5-fold runs are shown

	Mean	Standard Deviation
Recall	0.7691	0.0050
Precision	0.6905	0.0060
Accurcay	0.7119	0.0056

When compared to the ***SP-GO*** cross-validation results reported in Section 5.2.1, the ***SPonly*** classifiers have lower recall, precision and accuracy. The ***SP-GO*** *positives* consist mostly of articles collected from the GO database (85%). According to our Kullback-Leibler divergence analysis (Table 5.8 and Table 5.9), the term distribution over the articles from GO appears very different from the distributions characterizing our other datasets. The *positive* and the *negative* articles of the ***SP-GO*** dataset are easily distinguishable from each other, which contributes to the high performance observed in cross-validation studies. This is because of the significant difference between the term distribution within the *positive* articles and the other datasets, including the ***SP-GO*** *negatives*. When we remove the GO articles from the ***SP-GO*** *positives* to create the ***SPonly*** dataset, we remove the easily distinguishable articles. This explains why classifiers trained and tested on the ***SPonly*** dataset exhibit lower cross-validation performance than classifiers trained on the ***SP-GO*** dataset.

5.5.2 Test Results for Classifiers trained on the ***SPonly*** Dataset

In order to evaluate classifiers trained using the ***SPonly*** dataset, we used these classifiers to label each article in the ***Curated Journal***, ***Curated Swiss-Prot*** and ***CHAR*** datasets and recorded the recall, precision and accuracy. We tested the standard and only-present-terms classifiers with feature sets generated by the feature selection parameters listed in Table 5.10, with the exception of the document frequency cutoffs. In order to run fewer tests, we only examine using a minimum document frequency cutoff to exclude terms occurring in fewer than 3, 30, and 100 articles and a maximum document frequency cutoff to exclude terms occurring in more than 10%, 30% and 60% of the articles. The performance of classifiers trained on the ***SPonly*** dataset and tested on the ***Curated Journal***, ***Curated Swiss-Prot*** and ***CHAR*** datasets is much better than the performance reported in Table 5.5 for the ***SP-GO*** trained classifiers. Results over the default feature set are reported here and a full listing of the performance in each experiment is provided in Appendix D.2.

For the standard naïve Bayes classifiers, recall over the ***Curated Journal*** dataset was typically greater than 0.6 and approaches 0.74 on some feature sets, as can be seen in Appendix D.2. Precision and accuracy on this dataset were lower than recall, with values between 0.47 and 0.56. Results on the ***Curated Swiss-Prot*** dataset were quite high, often surpassing the 0.7 recall and 0.8 precision, specified as the required level of performance by CHAR curators. Recall on this dataset was typically between 0.66 and 0.73, precision between 0.83 and 0.88, and accuracy typically ranged from 0.69 to 0.74. Accuracy on

the ***CHAR*** validation set was typically between 0.65 and 0.71. Table 5.13 shows the results of a standard naïve Bayes classifier trained using the default feature set and tested over the curated and the ***CHAR*** datasets.

Table 5.13: Test results for a standard naïve Bayes classifier trained over the ***SPonly*** dataset and tested on the other datasets, using the default feature set

	Curated Journal	***Curated Swiss-Prot***	***CHAR***
Recall	0.7396	0.7037	
Precision	0.5221	0.8382	
Accuracy	0.5567	0.7189	0.7098

As shown in Appendix D.2, when the modified, only-present-terms calculation is employed, the performance of classifiers trained over the ***SPOnly*** dataset varies greatly depending on the feature set used. Recall on all datasets, as well as accuracy on the ***CHAR*** dataset, appear to be improved compared to classifiers that use the standard naïve Bayes calculation. Over the ***Curated Journal*** dataset, recall varies between 0.76 and 0.9, precision ranges from 0.46 to 0.5 and accuracy between 0.46 and 0.53. On the ***Curated Swiss-Prot*** dataset, recall is typically between 0.76 and 0.89, precision varies between 0.71 and 0.83 and accuracy ranges from 0.73 to 0.77. Accuracy on the CHAR validation set is between 0.77 and 0.91. The performance of an only-present-terms classifier trained on the ***SPonly*** dataset using the default feature set and tested on the curated and ***CHAR*** datasets is shown in Table 5.14.

Table 5.14: Test results for an only-present-terms naïve Bayes classifier trained using the ***SPonly*** dataset and tested on the other datasets, using the default feature set

	Curated Journal	***Curated Swiss-Prot***	***CHAR***
Recall	0.7917	0.7932	
Precision	0.4903	0.8159	
Accuracy	0.5123	0.749	0.8

Classifiers trained using the ***SPonly*** set outperform those trained using the ***SP-GO*** set on all other datasets, and achieve recall greater than 0.7 and precision greater than 0.8 on the ***Curated Swiss-Prot*** dataset, which surpasses the requirements set by the CHAR database curators. Performance using the only-present-terms calculation is better than that obtained by classifiers using the standard naïve Bayes calculation on the ***Curated Swiss-Prot*** and ***CHAR*** datasets. However, when we examine the effect of different feature sets on the only-present-terms classifier, the results fluctuate in unexpected ways. Performance was better when many rare and frequent terms were excluded from the feature set, and counting the last 20 abstract terms twice resulted in the best performance. These trends are unlike what we observed with classifiers trained on the ***SP-GO*** dataset and with the standard naïve Bayes classifier trained on the ***SPonly*** dataset.

Furthermore, it is possible that the improved recall and accuracy shown by the only-present-terms classifier compared to the standard classifier, when trained on the ***SPonly*** dataset, is a result of a bias towards classifying articles as *positive*. In cross-validation studies of classifiers trained on the ***SP-GO*** and ***SPonly*** datasets (Sections 5.2.1 and 5.5.1), we observe that the only-present-terms classifier tends to classify articles as *positive* more often than the standard classifier, resulting in improved recall at the cost of

lower precision and accuracy. Since the ***Curated Swiss-Prot*** dataset has more *positive* articles than *negatives*, and the ***CHAR*** dataset is composed entirely of *positive* articles, a bias toward labeling articles as *positive* improves both recall and accuracy over these datasets.

Chapter 6

Term Distribution Analysis

Classifiers trained on the ***SP-GO*** dataset misclassify the majority of the articles in the other datasets, which can be seen in Section 5.3.2. Classifiers trained on the ***SPonly*** dataset, do not meet the requirements specified by the CHAR curators when tested on the ***Curated Journal*** dataset, which can be seen in Section 5.6.2. In this chapter, we investigate the term distributions within the different datasets, and revisit the construction of the datasets, in order to explain these results.

The terms whose distribution is most different between the *positives* and *negatives* articles of a dataset are most predictive of the class and therefore have a large impact on the classifier. Furthermore, these terms can provide insight into the topics or trends that distinguish between the *positive* and *negative* articles within the dataset, which can help to explain classification performance. In order to determine the terms that best differentiate between the *positive* and *negative* articles of a dataset, we analyze the terms with the highest *Z-scores* (as defined in Section 4.2.2). High *Z-scores* indicate that the

difference between the probability of the term to occur in the *positive* articles and the probability of the term to occur in the *negative* articles is highly statistically significant.

First, we examine the ***SP-GO*** dataset by investigating its construction and by identifying terms with high *Z-scores* that may explain the performance of classifiers trained on the dataset. Next, we investigate the difference between the construction of the ***SPonly*** and the ***Curated Journal*** datasets and discuss the terms that may explain why classifiers trained on the ***SPonly*** dataset perform poorly on the articles of the ***Curated Journal*** dataset.

6.1 Construction and Term Distribution of the ***SP-GO*** Dataset

As shown in Section 5.3.2, classifiers trained using the ***SP-GO*** dataset misclassify the majority of the articles from the ***Curated Jounral***, ***Curated Swiss-Prot*** and ***CHAR*** datasets. By examining the Kullback-Leibler divergence between the term distributions of each dataset, we determined that the *positive* articles in the ***SP-GO*** set that were collected from GO are very different from the *positive* articles from the other sets (Sections 5.4.1 and 5.4.2). Moreover, when we removed these GO articles to create the ***SPonly*** dataset, the classifier's performance improved significantly. In this section, we further analyze the *positive* articles from GO in an attempt to explain why their term distribution is so different and why they had a negative impact on classification performance.

When building the ***SP-GO*** dataset, we collected all the PubMed references in GO that were labeled with an experimental evidence code (Section 3.3). There are six

experimental evidence codes, (EXP, IDA, IPI, IMP, IGI and IEP), and each code indicates the type of work or analysis reported in the associated reference article [19]. While the evidence codes are not directly associated with a single experimental technique, often there are only a few types of experiments that can provide the evidence reported by each code.

We examined the distribution of the different GO experimental evidence codes over the PubMed articles we collected and found that two of the six codes (IDA and IPI) account for roughly 70% of the articles. The *'Inferred from Direct Assay'* evidence code (IDA) is used to label references in which the function, process or component assigned to a gene product was determined through direct experimentation, such as an enzyme assay or a binding assay. The *'Inferred from Physical Interaction'* evidence code (IPI) is used to label references in which the described experiment demonstrates that the gene product interacts with another molecule, such as in a two-hybrid interaction or binding assay. IPI can be thought of as subtype of IDA, and is assigned when the molecule that interacts with the gene product can be included in the GO annotation using the 'with' field. Both of these evidence codes are strongly associated with binding assay experiments.

When examining a list of terms from the ***SP-GO*** dataset, sorted according to descending *Z-score*, several terms related to the IDA and IPI evidence codes appear overrepresented in the ***SP-GO*** *positives*, which are mostly articles from GO. Table 6.1 lists some of the IDA or IPI related terms. A full list of the 50 top scoring terms from the ***SP-GO*** dataset (according to the *Z-score*) can be found in Appendix E.1.

Table 6.1: IDA or IPI related terms that are overrepresented in the *positive* articles of the ***SP-GO*** dataset. *Positive* count indicates the number of *positive* articles that contain the term. *Negative* count indicates the number of *negative* articles that contain the term. Rank indicates the position of the term in the list sorted by *Z-score*. The suffix !T following a term indicates a title term. Terms are presented in their stemmed form

Term	*Z-score*	*Positive* count	*Negative* count	Rank
interact	31.63643	3244	1398	2
fission yeast	26.43377	1011	148	7
schizosaccharomyc pomb	25.13144	1012	181	12
interact!T	22.36503	1060	281	17
complex	22.32567	2521	1313	17
bind	20.60566	3766	2473	26

The terms interact, complex and bind are all overrepresented in the ***SP-GO*** *positives*, and are all strongly related to the binding assays that the IDA and IPI evidence codes describe. The high overrepresentation of yeast-related terms such as fission yeast and schizosaccharomyc pomb may also be related to the unevenly distributed GO evidence codes, as two-hybrid experiments are labeled as IPI by GO curators, and yeast is the most common organism in which two-hybrid experiments are performed.

It is likely that the *positive* part of the ***SP-GO*** dataset contains many articles that all describe results from the same type of experiments, leading to a classifier that labels articles reporting other types of experiments as *irrelevant*, regardless of their actual relevance to CHAR. This may contribute to the low recall and low overall accuracy shown by the ***SP-GO*** classifier over the curated and ***CHAR*** datasets in Section 5.2.2, and may explain why classification performance is improved by removing the GO articles.

6.2 Construction and Term Distribution of the ***SPonly*** Dataset

Classifiers trained using the ***SPonly*** dataset perform well on the ***CHAR*** validation set, and surpass the 0.7 recall and 0.8 precision requirements specified by CHAR curators on the ***Curated Swiss-Prot*** dataset (Section 5.5.2). However, performance on the ***Curated Journal*** dataset does not meet the CHAR curators' requirements, with precision and overall accuracy around 0.5. In order to explain the latter low performance, we first examine the construction of the ***Curated Journal*** dataset. Next, we investigate the 'scope' label that the Swiss-Prot database's curators have assigned to the articles in the ***SPonly*** dataset. Finally, we review the terms whose distributions are most different between the *positive* and *negative* articles in the ***SPonly*** dataset, and identify terms that may contribute to the low performance on the ***Curated Journal*** dataset.

The ***Curated Journal*** dataset is unique in its construction as its articles were not taken from public databases. We note that the ***SP-GO*** dataset was built by collecting articles from the GO and Swiss-Prot databases, the ***SPonly*** and the ***Curated Swiss-Prot*** datasets consist of articles from the Swiss-Prot database, and the ***CHAR*** dataset was created from articles in the CHAR database. In contrast, the ***Curated Journal*** dataset was constructed by having a curator manually examine every article from specific four journal issues. During the manual curation process, 96 of the articles were found to be *relevant* to the CHAR database, while 107 were *irrelevant*.

There is a large difference in the topics that may be present in the *negative* articles in the ***Curated Journal*** dataset compared to the *negative* articles in the ***Curated Swiss-***

Prot or in the ***SPonly*** dataset. The *negative* articles from all three datasets are *irrelevant* to the CHAR database because they do not contain experimental characterization of proteins. *Negative* articles from the ***Curated Swiss-Prot*** or from the ***SPonly*** dataset are, however, *relevant* to the Swiss-Prot database because they contain some other type of protein information; otherwise, they would not have been referenced by Swiss-Prot in the first place. Conversely, the *negative* articles in the ***Curated Journal*** dataset may not be *relevant* to any protein database. These articles could be about any topic that is published in the three journals they were collected from. The fact that the negative articles of the ***Curated Journal*** dataset do not necessarily discuss proteins may explain why the term distributions are relatively different from those of the other datasets according to our Kullback-Leibler divergence results (Tables 5.8 and 5.9). This difference may have also contributed to the low performance exhibited by classifiers when tested on the ***Curated Journal*** dataset (Tables 5.13 and 5.14).

The *negative* articles in the ***SP-GO*** and ***SPonly*** datasets were collected from Swiss-Prot (as described in Section 3.2). By locating GenBank sequences that lacked an 'experimental' flag, and mapping the GenBank sequences to identical sequences in Swiss-Prot, we identified Swiss-Prot entries that were unlikely to have been experimentally characterized. We collected all the articles referenced by these entries to build the ***SP-GO*** and ***SPonly*** *negative* sets. Such articles, which are associated with proteins sequences that have not been experimentally characterized, are likely to be early papers pertaining to the proteins, reporting their genomic and proteomic sequence.

The Swiss-Prot curators assign a 'scope' label to each article they reference. The 'scope' is a short identifier, indicating what type of information the article contributed to the Swiss-Prot entry. The ***SPonly*** *positive* articles all carry the 'CHARACTERIZATION' scope. Table 6.2 lists the 5 most common 'scope' labels for the *negative* articles from the ***SPonly*** dataset. Swiss-Prot curators often label articles with more than one scope, which is why the number of scope instances reported in the table exceeds the number of articles in the ***SPonly*** *negative* set.

Table 6.2: Top 5 Swiss-Prot scopes for the 1451 *negative* articles from the ***SPonly*** dataset. Count indicates the number of articles with the scope

Scope	Count
NUCLEOTIDE SEQUENCE [MRNA]	660
NUCLEOTIDE SEQUENCE [GENOMIC DNA]	387
TISSUE SPECIFICITY	342
FUNCTION	291
SUBCELLULAR LOCATION	175

As expected, the majority of the *negative* ***SPonly*** articles appear to report nucleotide sequence data. When we examine the terms that best differentiate between *positive* and *negative* articles of the ***SPonly*** dataset, very few of the 50 top-scoring terms (according to the *Z-score)* are overrepresented in the *negative* ***SPonly*** articles, compared to the *positives*. Of the terms that are overrepresented in the *negatives*, many pertain to genome or protein sequences. Table 6.3 lists these sequence-related terms. The 50 top-scoring terms (according to the *Z-score),* which differentiate between the *positive* and *negative* articles of the ***SPonly*** dataset, are presented Appendix E.2.

Table 6.3: Sequence-related terms that are overrepresented in the *negative* articles of the ***SPonly*** dataset. *Positive* count indicates the number of *positive* articles that contain the term. *Negative* count indicates the number of *negative* articles that contain the term. Rank indicates the position of the term in the list sorted by *Z-score*. The suffix !T following a term indicates a title term. Terms are presented in their stemmed form

Term	*Z-score*	*Positive* count	*Negative* count	Rank
genom!T	-8.52801	9	94	15
gene!T	-6.95649	313	480	31
region	-6.7497	301	461	34
intron	-5.8909	36	104	46
transcript	-5.87735	259	391	47

These terms are consistent with the hypothesis that the majority of the *negative* ***SPonly*** articles are early papers that report protein sequence. Other sequence-related terms, such as chromosom!T, genom sequenc!T, and cdna, are also more common in the *negative* ***SPonly*** articles than in the *positives* (with *Z-scores* ranks of 65, 97 and 142, respectively).

As discussed earlier, the *negative* articles of the ***Curated Journal*** dataset are unlike the *negative* articles from the other datasets because they may not be *relevant* to any protein database and, therefore, have a wide range of potential topics. Moreover, since the *negative* articles in the ***Curated Journal*** dataset are less likely to discuss sequence data than the *negative* articles in the ***SPonly*** dataset, classifiers trained on the ***SPonly*** dataset may erroneously label the *irrelevant*, non-sequence articles in the ***Curated Journal*** dataset as *positive*. Both of these factors are likely to have contributed to the poor performance demonstrated by classifiers trained on the ***SPonly*** dataset, when applied to the ***Curated Journal*** dataset.

Chapter 7

Conclusions and Future Work

We have trained and tested naïve Bayes classifiers with the goal of identifying journal articles that potentially contain relevant protein characterization information. The articles were classified based on a multi-variate Bernoulli representation of their titles and abstracts. We examined features selected based on the document frequency of each term and on the *Z-score*. We evaluated using unigrams and bigrams as terms, as well as counting title and late abstract terms twice. We also explored a method to identify species-specific terms and remove or replace them with a common placeholder, in order to avoid future classification bias toward certain species. We evaluated both the standard naïve Bayes classifier and a modified version, which only considers the terms present in the article being classified.

We constructed four datasets in order to train and test the classifiers. The ***SP-GO*** dataset was created by collecting articles from the GO and Swiss-Prot public databases based on how they were labeled by the public database curators. The ***Curated Journal*** dataset was created by manually labeling articles collected from four issues of three journals. The ***Curated Swiss-Prot*** dataset was created by manually labeling articles collected from the Swiss-Prot database. Interestingly, we found that for 84% of the

articles that were manually labeled, it was possible to determine if the article was *relevant* or *irrelevant* based on its title and abstract alone. While the pertinent information is most likely to be found in the full text of the article, it is important to note that the relevance can be determined in most cases from the title and abstract alone.

Finally, the ***CHAR*** dataset was created by collecting articles from the CHAR database, and only contains *relevant* examples.

After considering our initial results, we determined, by examining the Kullback-Leibler divergence between the term distributions of our various datasets, that the *relevant* articles we collected from the GO database were very different from *relevant* articles in the other datasets. We created the ***SPonly*** dataset by removing the articles originating from GO out of the ***SP-GO*** dataset.

7.1 Contributions

In this thesis we have made the following contributions, which have implications for identification of protein characterization articles, as well as for more general text classification tasks:

- We evaluated the impact on classification performance of using title terms, bigrams and late abstract terms, as well as several feature selection methods. We also compared a standard naïve Bayes classifier to our modified, only-present-terms classifier. The modified naïve Bayes classifier shows slightly better performance than the standard classifier in a number of situations, however, the

increase in performance may be the result of a bias toward labeling articles as *relevant*.

- We observed that classifiers trained on the ***SP-GO*** dataset incorrectly classify the majority of the articles in the other datasets, which was an unexpected result given the size of the dataset and the careful consideration applied when creating it. From this result, we can conclude that using information from large public resources to build a dataset of many examples is not necessarily an effective approach, despite the fact that the definitions of the curation labels seem consistent with the criterion of relevance to CHAR.
- In contrast, we demonstrate that classifiers trained on each of the small, manually curated datasets performed well when tested on the other curated dataset and the ***CHAR*** dataset, despite the fact that these datasets only contain a few hundred example articles. These results indicate that creating a training dataset by manual curation may be a viable option for text classification tasks where other sources of labeled examples are not available.
- Finally, we show that classifiers trained on the ***SPonly*** dataset show much better performance than those trained on the ***SP-GO*** dataset over the other datasets. Over the ***Curated Swiss-Prot*** and ***CHAR*** datasets, the performance of ***SPonly*** trained classifiers surpasses the 0.8 recall and 0.7 precision requirements stipulated by the curators of the CHAR database.

7.2 Future Work

The ***SPonly*** trained classifier surpasses the CHAR database curator's specifications for a classifier to be useful when tested over the ***Curated Swiss-Prot*** and ***CHAR*** datasets. However, the classifier does not meet the specified thresholds when applied to the ***Curated Journal*** dataset. Several steps could be taken to improve performance in general, and especially over the ***Curated Journal*** dataset.

We expect the largest improvements to be gained by constructing a new training dataset. As we discussed in Section 6.2, the *irrelevant* articles in the ***Curated Journal*** dataset discuss a wide range of topics and may not be *relevant* to any protein databases, in contrast to the *irrelevant* articles from the ***SPonly*** and ***Curated Swiss-Prot*** databases, which are *relevant* to the Swiss-Prot database. By manually labeling articles collected from journals that cover many biomedical topics in addition to protein characterization, a classifier could be trained that is more robust and better suited to classifying articles from general sources. It is unclear how many additional articles would need to be labeled in order to achieve a significant gain in performance; this may not be a viable option if the number of articles needed is prohibitively high. However, we have shown that small datasets can be used to train effective classifiers (Section 5.3), and the new dataset could be supplemented by our existing manually labeled articles as well as a sampling of the articles we collected from public resources.

Alternatively, we could limit the sources of the articles we classify in order to exclude articles that are very unlikely to be relevant to CHAR and focus on articles the

current system classifies reliably. The current classifier performs well on articles that have been used as references in the Swiss-Prot database, and it is possible that it would also perform well on articles that have been referenced in any protein database. Moreover, there may be keyword searches or other methods that could exclude articles from general sources that the classifier is likely to mislabel because they discuss topics *irrelevant* to any protein database. Since the classifier exceeds the specifications defined by CHAR's curators on specific types of articles, namely, articles that are potentially relevant to CHAR, the next step could be the coarser task of identifying articles that the classifier will perform well on and excluding articles that are very different than those sought after by CHAR's curators.

Classifier performance may be further improved by identifying gene and protein names and replacing them with common placeholders, much like the approach we take with species-specific terms in Section 4.2.3. The presence of a gene or a protein name in an article can be a powerful feature for class prediction. However, while named entity recognition in text has been studied in the biomedical domain [71, 67], it is not a trivial process.

Another possible way to improve classifier performance would be to explicitly use expert knowledge to increase the weight of terms known to be indicative of protein characterization experiments. Terms such as the units or variables used to describe experiment results, the reagents or equipment used for relevant procedures, or even the names of the experiments themselves could be highly indicative, and increasing their

weight could increase performance. However, the downside to using expert knowledge to modify machine learning techniques is that, in the future, if the set of terms that are highly indicative of relevance changes, the experts must be consulted again and the system must be updated accordingly. Conceptual drift, the process by which the terms used to discuss topics change over time [7], can clearly be seen in the ***SP-GO*** dataset (Section 3.3). Modifying the classifier to rely on terms that, currently, are indicative of relevance may become an issue later on.

Over the course of this work, we only employed one term weighting scheme (namely, multi-variate Bernoulli), one kind of classifier (naïve Bayes), and only a few common feature selection approaches (document frequency and *Z-score*). It is possible that other approaches and techniques may perform better on this task. Term weighting schemes such as term frequency by inverse document frequency [31], classifiers such as Maximum Entropy [46] or Support Vector Machines, and feature selection approaches such as Information Gain [11] or the χ^2-*Statistic* [69] could be evaluated to investigate their impact on performance.

A system capable of extracting protein characterization information would be a logical extension to the text classification we presented in this thesis. This would be much like the system proposed by Schmeier et al. [56], which extracts kinetic parameters, and the system described by Donaldson et al. [11], which identifies potential protein-protein interactions. An information extraction approach could collect potentially relevant sentences from the text that discuss the results of the protein characterization

experiments and present them to database curators. A more sophisticated system might be able to identify the associations between protein names and biological functions and, therefore, infer the characterization information being reported in the article. These, currently hypothetical, systems could save database curators effort by providing the curator with the relevant sentences from the article, or even populating the fields of the database itself. The current classifier would be the first step; identifying relevant articles so that the information extraction system could act upon them.

We expect such modifications and extensions to further improve the performance of our classifiers and make them more useful to protein database curators. Our text classification system, in its current form, can identify relevant articles, suggesting the presence of protein characterization experiments, and performs especially well over articles from the Swiss-Prot database. It can be used to reduce the amount of curator time needed to populate a database of characterized proteins.

Bibliography

[1] S.F. Altshul, W. Gish, W. Miller, E.W. Myers and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, vol. 215, pp. 403-410, 1990.

[2] C. Apté, F.J. Damerau, and S.M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems* vol. 12, no. 3, pp. 233-251, 1994.

[3] S. Brady. Text-based prediction of protein subcellular location. MSc Thesis, School of Computing, Queen's University, Kingston, ON, 2007.

[4] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology,* vol. 268, pp. 78-94, 1997.

[5] C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998.

[6] S.B. Carroll, B. Prud'homme and N. Gompel. Regulating Evolution . *Scientific American.* New York: Scientific American Inc. pp. 60–67, May 2008.

[7] A.M. Cohen, R.T. Bhupatiraju, W.R. Hersh. Feature generation, feature selection, classifiers, and conceptual drift for biomedical document triage. *Proceedings of the Thirteenth Text Retrieval Conference: TREC 2004,* Gaithersburg, MD, National Institute of Standards and Technology, 2004.

[8] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic indexing. *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[9] A.L. Delcher, K.A. Bratke, E.C. Powers and S.L. Salzberg. Identifying bacterial genes and endosymbiont DNA with Glimmer. *Bioinformatics*, vol. 23, no. 6, pp. 673-679, 2007.

[10] R.C. Deonier, S. Tavaré, M.S. Waterman. Chapter 6: Sequence Alignment and Chapter 7: Rapid Alignments Methods: FASTA and BLAST. *Computational Genome Analysis*. New York, NY: Springer Science + Business Media, LLC, 2005.

[11] I. Donaldson, J. Martin, B. de Bruijn, C. Wolting, V. Lay, B. Tuekam, S. Zhang, B. Baskin, G. D. Bader, K. Michalickova, T. Pawson and C.W.V. Hogue. PreBIND and Textomy – mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics*, vol. 4, no. 11, 2003.

[12] R.O. Duda, P.E. Hart, and D.G. Stork. Chapter 2: Bayesian Decision Theory. *Pattern Classification*, 2nd ed., New York, NY: John Wiley & Sons, Inc. 2001.

[13] R. Durbin, S. Eddy, A. Krogh, G. Mitchison. Chapter 5: Profile HMMs for Sequence Families. *Biological Sequence Analysis.* UK: Cambridge University Press, 1998

[14] FlyBase: A Database of Drosophila Genes & Genomes. http://flybase.org/, Accessed: Sept. 26, 2009.

[15] J.E. Freund. Chapter 14: Tests of Hypotheses Based on Count Data. *Modern Elementary Statistics*, 11th ed., New Jersey: Pearson Education, Inc., 2004.

[16] J. Fürnkranz. A study using n-gram features for text categorization. *Technical Report OEFAI-TR-98-30, Austrian Institute for Artificial Intelligence*. 1998.

[17] L.A. Galli-Taliadorosa, J.D. Sedgwicka, S.A. Woodb and H. Körner. Gene knock-out technology: a methodological overview for the interested novice. *Journal of Immunological Methods,* vol. 181, no. 1, pp. 1-15, 1995.

[18] The Gene Ontology Project. http://www.geneontology.org/, Accessed: Mar. 16, 2009.

[19] The Gene Ontology Project: Guide to GO Evidence Codes. http://www.geneontology.org/GO.evidence.shtml, Accessed: Mar. 16, 2009.

[20] D.H. Haft, J.D. Selengut, and O. White. The TIGRFAMs database of protein families. *Nucleic Acids Research,* vol. 31, no. 1, pp. 371-373, 2003.

[21] W. Hersh, and R.T. Bhupatiraju. TREC Genomics Track Overview. *Proceedings of the Twelfth Text Retrieval Conference (TREC 2003),* Gaithersburg, MD: National Institute for Standards & Technology, 2003.

[22] W.R. Hersh, R.T. Bhupatiraju, L. Ross, P. Johnson, A.M. Cohen, D.F. Kraemer. TREC 2004 Genomics Track Overview. *Proceedings of the Thirteenth Text Retrieval Conference (TREC 2004),* Gaithersburg, MD: National Institute for Standards & Technology, 2004.

[23] W. Hersh, A. Cohen, J. Yang, R.T. Bhupatiraju, P. Roberts, M. Hearst. TREC 2005 Genomics Track Overview. *Proceedings of the Fourteenth Text Retrieval Conference (TREC 2005),* Gaithersburg, MD: National Institute for Standards & Technology, 2005.

[24] W. Hersh, A.M. Cohen, P. Roberts, H.K. Rekapalli. TREC 2006 Genomics Track Overview. *Proceedings of the Fifteenth Text Retrieval Conference (TREC 2006)*, Gaithersburg, MD: National Institute for Standards & Technology, 2006.

[25] W. Hersh, A. Cohen, L. Ruslen, P. Roberts. TREC 2007 Genomics Track Overview. *Proceedings of the Sixteenth Text Retrieval Conference (TREC 2007),* Gaithersburg, MD: National Institute for Standards & Technology, 2007.

[26] X. Huang, M. Zhong and L. Si. York University at TREC 2005: Genomics Track. *Proceedings of the Fourteenth Text Retrieval Conference (TREC 2005)*, Gaithersburg, MD: National Institute for Standards & Technology, 2005.

[27] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of the 10th European Conference on Machine Learning (ECML-98),* pp. 137–142, 1998.

[28] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence,* San Francisco, CA: Morgan Kaufmann, pp. 1137–1143, 1995.

[29] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79-86, 1951.

[30] S.L.Y. Lam & D.L. Lee. Feature Reduction for Neural Network Based Text Categorization. *Proceedings of the 6th IEEE International Conference on Database Advanced Systems for Advanced Application (DASFAA-99),* pp. 195–202, 1999.

[31] M. Lan, C. Tan, H. Low, and S. Sung. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. *Posters Proceedings of the 14th International World Wide Web Conference,* pp. 1032-1033. 2005.

[32] B. Larsen, and C. Aone. Fast and effective text mining using linear-time document clustering. *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, pp. 16–22, 1999.

[33] J.E. Leonard, J.B. Colombe and J.L. Levy. Finding relevant references to genes and proteins in Medline using a Bayesian approach. *Bioinformatics,* vol. 18, no. 11, pp. 1515-1522, 2002.

[34] D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. *Proceedings of the 15th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR-92)*, pp. 37-50, 1992.

[35] D. Lewis. Naïve (Bayes) at forty: the independence assumption in information retrieval. *Proceedings of the 10th European Conference on Machine Learning (ECML'98)*, pp. 4-15, 1998.

[36] Y. Li and A. Jain. Classification of text documents. *The Computer Journal*, vol. 41, no.8, pp. 537-546, 1998.

[37] G. MacBeath, S.L. Schreiber. Printing Proteins as Microarrays for High-Throughput Function Determination. *Science,* vol. 289, no. 5485, pp. 1760-1763, 2000.

[38] R. Madupu. Personal Communication., Informatics Department, J. Craig Venter Institute, 2008-2009.

[39] A. Mak, J. Allingham, Z. Jia. Course Lectures: Protein Structure and Function Biochemistry 410, Queen's University, Kingston, ON. Sept-Dec, 2006.

[40] E. R. Mardis. Next generation DNA sequencing methods. *Annual Review of Genomics and Human Genetics,* vol. 9, pp. 387–402, 2008.

[41] A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. *Proceedings of the Association for the Advancement of Artificial Intelligence Workshop on Learning for Text Categorization (AAAI'98),* pp. 41-48, 1998.

[42] MGI – Mouse Genome Informatics. http://www.informatics.jax.org/, Accessed: Nov. 1, 2009.

[43] NCBI – GenBank. http://www.ncbi.nlm.nih.gov/Genbank/, Accessed: Nov. 1, 2009

[44] NCBI-GenBank Flat File Release 172.0: Distribution Release Notes. ftp://ftp.ncbi.nih.gov/genbank/release.notes/gb172.release.notes, Accessed: Sept 30. 2009.

[45] NCBI Taxonomy Database. http://www.ncbi.nlm.nih.gov/taxonomy, Accessed: May 21, 2009.

[46] K. Nigam, J. Lafferty, and A. McCallum. Using Maximum Entropy for text classification. *Proceedings of the Workshop on Machine Learning for Information Filtering (IJCAI-99),* pp. 61-67, 1999.

[47] K. Nigam, A.K. McCallum, S. Thrun and T.M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning,* vol. 39, no. 2/3, pp. 103-134, 2000.

[48] F. Pan. Multi-Dimensional Fragment Classification in Biomedical Text. MSc thesis, Queen's University, Kingston, ON, Canada, 2006.

[49] M.F. Porter. An algorithm for suffix stripping. *Program*, vol. 14, no. 3, pp. 130-137, 1980.

[50] PubMed. http://www.ncbi.nlm.nih.gov/pubmed, Accessed: Mar. 16, 2009.

[51] Y. Regev, M. Finkelstein-Landau, R. Feldman, M. Gorodetsky, X. Zheng, S. Levy, R. Charlab, C. Lawrence, R.A. Lippert, Q. Zhang and H.Shatkay. Rule-based Extraction of Experimental Evidence in the Biomedical Domain – the KDD Cup 2002 (Task 1). *ACM SIGKDD Explorations Newsletter*, vol. 4 no. 2, pp. 90-92, 2002.

[52] C. Van Rijsbergen. Chapter 7: Evaluation. *Information Retrieval*, 2nd ed. London, UK: Butterworths. 1979.

[53] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, vol. 27, no. 3, pp. 129-146, 1976.

[54] M. Sahami. Using machine learning to improve information access. Ph.D. thesis, Stanford University, California, USA, 1998.

[55] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, vol. 24, no. 5, pp. 513-523, 1988.

[56] S. Schmeier, J. Hakenberg, A. Kowald, E. Klipp, and U. Leser. Text mining for systems biology using statistical learning methods. *Proceedings of the Workshop des Arbeitskreises Knowledge Discovery*, pp. 125-129, 2003.

[57] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys,* vol. 34, no. 1, pp. 1–47, 2002.

[58] SGD – Saccharomyces Genome Database. http://www.yeastgenome.org/, Accessed Nov. 1, 2009.

[59] H. Shatkay, R. Feldman. Mining the Biomedical Literature in the Genomic Era: An Overview. *Journal of Computational Biology*, vol. 10, no. 6, pp. 821-855, 2003.

[60] L. Stein. Genome Annotation: From Sequence to Biology. *Nature Reviews Genetics*, vol. 2, pp. 493–503, 2001.

[61] Swiss-Prot Protein Knowledgebase. http://ca.expasy.org/sprot/, Accessed: Mar. 16, 2009.

[62] Swiss-Prot Protein Knowledgebase: Release Notes for UniProtKB Release 14.0 of 22-Jul-2008. http://www.expasy.ch/txt/old-rel/relnotes.56.htm, Accessed: Sept. 29, 2009.

[63] TREC Genomics Track. http://ir.ohsu.edu/genomics/, Accessed: Sept 26, 2009.

[64] The J. Craig Venter Institute: JCVI Annotation Service Overview. http://www.jcvi.org/cms/research/projects/annotation-service/overview/, Accessed: Aug. 12, 2009.

[65] D. Voet and J.G. Voet. *Biochemistry*, 3rd Ed. New Jersey: John Wiley & Sons, Inc., 2004.

[66] R.E. Walpole, R.H. Myers, S.L. Myers. Section 10.12 Two Samples: Tests on Two Proportions. *Probability and Statistics for Engineers and Scientists*. Prentice-Hall, pp. 333-334, 1998.

[67] J. Wilbur, L. Smith and T. Tanabe. BioCreative 2. Gene MentionTask. *Proceedings of the Second BioCreAtIvE Challenge Workshop,* pp. 7-16, 2007.

[68] K. Wüthrich. Protein Structure Determination in Solution by NMR Spectroscopy. *The Journal of Biological Chemistry,* vol. 265, no. 36, pp. 22059-22062, 1990.

[69] Y. Yang, J.O. Pedersen. A Comparative Study on Feature Selection in Text Categorization. *Proceedings of the 14th International Conference on Machine Learning* (*ICML-97*), pp. 412-420, 1997.

[70] A. Yeh, L. Hirschman and A. Morgan. Background and overview for KDD Cup 2002 task 1: information extraction from biomedical articles. *ACM SIGKDD Explorations Newsletter,* vol. 4, no. 2, pp. 87-89, 2002.

[71] A. Yeh, A. Morgan, M. Colosimo and L. Hirshman. Overview of BioCreAtIvE: Critical assessment of information extraction for biology. *BMC Bioinformatics*, vol. 6 (Suppl 1):S2, 2005.

[72] Y. Zhao, G. Karypis. Hierarchical Clustering Algorithms for Document Datasets. *Data Mining and Knowledge Discovery*, vol. 10, pp. 141–168, 2005.

Appendix A

Additional Biology Background

A.1 Automated Genome Annotation

In this section we discuss two methods for automated genome annotation. These methods are applied in order to label a new protein with information borrowed from other proteins that are likely to share the same function and have already been characterized. Such methods are necessary because the rate in which genomes are sequenced is much greater than the rate in which individual proteins can be experimentally characterized.

Two proteins that have evolved from a common ancestor are called homologs. If the proteins diverged recently, they share a similar sequence and, often, both proteins possess the same function. There are conditions where proteins that appear to be homologous may not share the same function, but, for brevity, we will not discuss them here. In homology-based genome annotation, the novel protein's sequence is searched against databases of protein sequences that have already been annotated. Then, if a protein sequence is found with a sufficiently high similarity to the novel protein's sequence, the novel protein can be annotated with the function of the similar protein. Algorithms have been developed in order to efficiently search through databases of

protein sequences. The most common is BLAST[5] and its derivatives, which find local alignments between the sequences and then determine a *p*-value for the alignment in order to rank the results [10]. By performing a BLAST search against characterized protein sequence databases, researchers can locate proteins that are homologous to the new protein in order to annotate it [60].

A protein's amino acid sequence defines its structure and function, but not all amino acids contribute equally. For a protein to conform to the correct shape and have the right function, certain amino acids in certain positions may be vital if they cause an important fold or interact with a target molecule. Other amino acids may be more interchangeable if they are, perhaps, simply spacers that connect the important fold to the interaction site. By studying protein families – sets of proteins with similar structure and the same function that are collected from different organisms – researchers are able to determine the most important amino acids by observing which ones have been conserved across the family. Statistical models have been built to represent protein families [13, 20]. If a new protein sequence fits the model of a protein family well, it indicates that the protein has the correct amino acids in the right places and, therefore, is likely to share the same function as the proteins in the family. Genome annotation can be performed for many sequences by comparing the new sequence to models of protein families and, if the sequence matches a model well enough, assigning the function of the best match to the new protein [60].

---

[5] BLAST stands for *Basic Local Alignment Search Tool*.

A.2 Protein Characterization

In this section we briefly describe several protein characterization techniques. Each technique can take many months to perform, and the results are typically published in scientific journals. Articles that discuss results from any of these techniques would be relevant to a characterized protein database.

The most direct way to study a protein is to isolate it. However, given the fact that the cell contains thousands of different proteins and proteins can be denatured (altered so that they lose their structure and function) by various chemicals and conditions, protein purification is not an easy task. One way to increase the amount of a specific protein in a cell is to insert its gene to the genome of a bacteria, often *E. coli*, so that the cell produces very high levels of the protein [39]. The downside to this approach is that the protein may fold differently than it would in cells from its original species, resulting in a different structure and, therefore, a different function than expected.

Methods such as chromatography or gel electrophoresis are capable of sorting proteins based on the various physical and chemical attributes they possess based on their amino acid composition. Protein chromatography is a technique where proteins are dissolved in a liquid and then percolated through a vertical column filled with a stationary, porous substance [65]. The mixture of proteins can be sorted by how quickly they move through the column. Depending on the liquid and stationary substances used, the proteins can be sorted by their size, solubility in water, pH, ionic charge, or their affinity for a molecule of interest.

Gel electrophoresis works in a similar manner, but instead of the protein mixture being pulled by gravity through a column, it is pulled by an electric field through a gel [39]. Using this technique, proteins can be separated by size and by their ionic charge in two dimensions along one flat gel, and any proteins of interest can be cut from the gel and further analyzed.

Once a protein has been purified, many different experiments can be performed. The protein can be sequenced in order to identify it using a technique called mass spectrometry. A common mass spectrometry approach, known as MALDI-TOF MS[6], involves fragmenting the protein into small pieces and then measuring their mass by timing their flight through a vacuum [39].

Protein microarrays are glass plates spotted with hundreds of microscopic dots, each one containing a different type of molecule, such as proteins, antibodies, or DNA sequences [37]. By washing the plate with the purified protein, researchers can determine which molecules the protein interacts with.

A highly purified protein's structure can be determined experimentally in two ways. X-ray crystallography requires that the protein be completely purified so that a protein crystal forms. By analyzing the diffraction pattern produced by x-rays passing through the protein researchers are able to determine its three dimensional structure [39]. Nuclear magnetic resonance spectroscopy is another method to determine the structure of

---

[6] MALDI-TOF MS is the acronym for matrix-assisted laser desorption/ionization time of flight mass spectrometry.

a protein. It uses very strong electromagnets to produce a field that excites certain atoms in the protein and then detect their position [68]. A protein's structure can reveal much about how it functions.

Another way to investigate a protein's function is to remove the corresponding gene from an organism's genome and then observe how the mutated organism differs from an unmodified organism. This technique is known as a *gene knock*out experiment [17]. Often no difference will be observable between the normal and mutant organisms, or the organism will fail to survive with the mutated genome. However, occasionally a difference will present which is indicative of the function of the protein which is coded for in the knocked-out gene. For example, if a specific molecule abnormally builds up within the mutant cells then it is likely that the knocked-out gene's protein is responsible for acting on that molecule. Experiments where the expression of a gene is increased instead of removed are also performed, and observing the effect of an abundance of the protein can also indicate its function.

Cells contain a number of compartments known as subcellular locations. Knowing where in the cell a protein is located can provide clues towards understanding the roll the protein serves within the cell. Protein subcellular location can be determined by fluorescence microscopy, a technique where the protein of interest is made to fluoresce light so that its position within the cell may be observed [39]. A common method of adding fluoresce to a protein is to fusing a Green Fluorescent Protein (GFP) to it by appending the protein's gene with the GFP gene.

Another fluorescence technique, which does not require modification of the protein being studied, is to expose a mammalian species, often a rabbit or goat, to the protein of interest [39]. The mammal's immune system will produce antibodies that bind specifically to the protein. Then, by adding fluorescence to the antibodies and introducing them into the cell, the antibodies will bind to the protein of interest, revealing its location.

The final type of protein characterization technique we describe consists of experiments capable of indicating interaction between two proteins. Proteins interact for a variety of reasons: to bind together to form large complexes, to transport proteins or other molecules, or to effect a chemical change in one of the proteins in order to modify its behavior, for example [65]. Two-hybrid experiments, which are often carried out in yeast cells, can identify protein-protein interactions. The yeast's genome is modified so that an indicator gene, such as a gene vital to creating an essential nutrient, can only be expressed if the two proteins of interest bind together. The yeast is grown in an environment that lacks the essential nutrient and whether the yeast is able to survive or not is the indicator of whether the two proteins interact [39].

Other methods of determining if two proteins interact include protein microarrays, which we discussed above, and fluorescence resonance energy transfer experiments, where the two proteins of interest are modified so that if they interact, they bring two different fluorescent proteins into close enough proximity that they share energy and give off light to indicate the interaction [39]. By repeating these types of experiments with

different proteins, researchers can discover new protein interactions and create protein interaction networks to help understand the system of proteins that drives the cell.

Appendix B

Full Results for Classifiers trained on the ***SP-GO*** Dataset

B.1 Cross-Validation Results

B.1.1 Effects of Filtering terms by Document Frequency

When using either the standard naïve Bayes classifier, which can be seen in Table B.1.1.1, or the modified, only-present-terms classifier, shown in Table B.1.1.2, and varying the document frequency cutoffs, performance appears to increase slightly as more terms are involved in the classification, with the exception of the case where the minimum cutoff is set to 3. When the minimum document frequency cutoff is 10, performance is better than when the default cutoff value, 3, is used. It appears that including terms that occur in fewer than 10 articles causes the classifier to mislabel articles.

B.1.1.1 Standard Calculation

The effects of filtering terms by document frequency on the performance of classifiers trained on the ***SP-GO*** dataset and using the standard calculation. Min indicates the minimum number of articles a term must occur in for it to be included in the feature set. Max indicates the maximum number of articles a term may occur in for it to be included in the feature set. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

		Min 100		Min 30		Min 10		Min 3	
Max		Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
10%	Recall	0.7368	0.0012	0.7400	0.0009	0.7392	0.0011	0.7354	0.0013
	Precision	0.7228	0.0006	0.7337	0.0007	0.7380	0.0011	0.7324	0.0016
	Accuracy	0.7275	0.0007	0.7361	0.0007	0.7388	0.0010	0.7338	0.0014
20%	Recall	0.7369	0.0009	0.7400	0.0008	0.7372	0.0010	0.7333	0.0016
	Precision	0.7245	0.0007	0.7333	0.0006	0.7387	0.0006	0.7332	0.0011
	Accuracy	0.7287	0.0006	0.7358	0.0005	0.7386	0.0006	0.7337	0.0011
30%	Recall	0.7375	0.0007	0.7400	0.0007	0.7369	0.0007	0.7347	0.0019
	Precision	0.7244	0.0004	0.7335	0.0006	0.7382	0.0009	0.7338	0.0012
	Accuracy	0.7288	0.0004	0.7359	0.0004	0.7382	0.0007	0.7345	0.0013
40%	Recall	0.7359	0.0009	0.7388	0.0008	0.7361	0.0010	0.7336	0.0011
	Precision	0.7253	0.0003	0.7334	0.0006	0.7385	0.0006	0.7337	0.0006
	Accuracy	0.7290	0.0003	0.7355	0.0006	0.7381	0.0006	0.7341	0.0007
60%	Recall	0.7356	0.0008	0.7383	0.0012	0.7360	0.0012	0.7333	0.0012
	Precision	0.7248	0.0005	0.7332	0.0007	0.7383	0.0011	0.7343	0.0014
	Accuracy	0.7285	0.0006	0.7352	0.0007	0.7380	0.0009	0.7343	0.0012

B.1.1.2 Only-Present-Terms Calculation

The effects of filtering terms by document frequency on the performance of classifiers trained on the ***SP-GO*** dataset and using the only-present-terms calculation. Min indicates the minimum number of articles a term must occur in for it to be included in the feature set. Max indicates the maximum number of articles a term may occur in for it to be included in the feature set. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

		Min 100		Min 30		Min 10		Min 3	
Max		Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
10%	Recall	0.7461	0.0007	0.7545	0.0010	0.7542	0.0010	0.7476	0.0013
	Precision	0.7198	0.0008	0.7264	0.0006	0.7308	0.0010	0.7248	0.0008
	Accuracy	0.7282	0.0006	0.7355	0.0006	0.7386	0.0009	0.7323	0.0007
20%	Recall	0.7322	0.0010	0.7421	0.0012	0.7445	0.0009	0.7429	0.0018
	Precision	0.7264	0.0006	0.7317	0.0009	0.7349	0.0011	0.7293	0.0010
	Accuracy	0.7286	0.0005	0.7354	0.0006	0.7384	0.0009	0.7339	0.0012
30%	Recall	0.7289	0.0007	0.7412	0.0010	0.7426	0.0007	0.7417	0.0015
	Precision	0.7287	0.0006	0.7331	0.0005	0.7361	0.0007	0.7300	0.0010
	Accuracy	0.7292	0.0006	0.7361	0.0005	0.7386	0.0006	0.7341	0.0011
40%	Recall	0.7259	0.0009	0.7382	0.0009	0.7406	0.0008	0.7404	0.0016
	Precision	0.7301	0.0007	0.7343	0.0006	0.7369	0.0004	0.7302	0.0011
	Accuracy	0.7291	0.0007	0.7359	0.0005	0.7385	0.0005	0.7338	0.0010
60%	Recall	0.7257	0.0006	0.7380	0.0009	0.7408	0.0012	0.7412	0.0013
	Precision	0.7304	0.0007	0.7345	0.0007	0.7374	0.0008	0.7296	0.0012
	Accuracy	0.7293	0.0005	0.7360	0.0006	0.7389	0.0008	0.7336	0.0008

B.1.2 Effects of Filtering terms by *Z-Score*

For the standard naïve Bayes classifier, when we vary the *Z-score* threshold, recall is highest at $Z = 2$ and precision and accuracy are both highest at $Z = 3$, as shown in Table B.1.2.1. Table B.1.2.2 gives the results we observed when using the only-present-terms classifier. Performance improves as we increase the *Z-score* threshold, with the highest precision and accuracy for any ***SP-GO*** trained classifier occurring when $Z = 5$.

B.1.2.1 Standard Calculation

The effects of filtering terms by *Z-score* on the performance of classifiers trained on the ***SP-GO*** dataset and using the standard calculation. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	No *Z-Score* filtering		Z = 2		Z = 3	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7333	0.0012	0.7411	0.0014	0.7378	0.0006
Precision	0.7343	0.0014	0.7331	0.0013	0.7363	0.0007
Accuracy	0.7343	0.0012	0.7360	0.0011	0.7372	0.0005

	Z = 4		Z = 5	
	Mean	Stdev	Mean	Stdev
Recall	0.7380	0.0008	0.7364	0.0007
Precision	0.7316	0.0007	0.7277	0.0006
Accuracy	0.7340	0.0006	0.7308	0.0005

B.1.2.2 Only-Present-Terms Calculation

The effects of filtering terms by *Z-score* on the performance of classifiers trained on the ***SP-GO*** dataset and using the only-present-terms calculation. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	No *Z-Score* filtering		Z = 2		Z = 3	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7412	0.0013	0.7405	0.0012	0.7368	0.0014
Precision	0.7296	0.0012	0.7314	0.0010	0.7416	0.0006
Accuracy	0.7336	0.0008	0.7347	0.0009	0.7404	0.0007

	Z = 4		Z = 5	
	Mean	Stdev	Mean	Stdev
Recall	0.7383	0.0006	0.7365	0.0009
Precision	0.7464	0.0005	0.7507	0.0015
Accuracy	0.7441	0.0005	0.7463	0.0011

B.1.3 Effects of Removing or Replacing Species-Specific terms

When using a standard naïve Bayes classifier (Table B.1.3.1) or when using the only-present-terms classifier (Table B.1.3.2) the results indicate that removing or replacing species-specific terms leads to a slightly lower performance than when these terms are included, which is to be expected. This is because, as stated in Section 4.2.3, some

species-specific terms are overrepresented in the *positive* or *negative* portions of the datasets, and therefore can be effective predictors of the class label. Removing or replacing species-specific terms sacrifices this effectiveness in order to avoid future bias where articles may be classifier based on the presence or absence of certain species-specific terms rather than whether or not they discuss protein characterization.

B.1.3.1 Standard Calculation

The effects of removing or replacing species-specific terms on the performance of classifiers trained on the ***SP-GO*** dataset and using the standard calculation. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	Replace species-specific terms		Remove species-specific terms	
	Mean	Stdev	Mean	Stdev
Recall	0.7252	0.0017	0.7267	0.0009
Precision	0.7251	0.0012	0.7218	0.0008
Accuracy	0.7255	0.0012	0.7237	0.0008

B.1.3.2 Only-Present Terms Calculation

The effects of removing or replacing species-specific terms on the performance of classifiers trained on the ***SP-GO*** dataset and using the only-present-terms calculation. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	Replace species-specific terms		Remove species-specific terms	
	Mean	Stdev	Mean	Stdev
Recall	0.7341	0.0016	0.7453	0.0018
Precision	0.7196	0.0010	0.7129	0.0010
Accuracy	0.7244	0.0011	0.7230	0.0010

B.1.4 Effects of Modifying the Term Set

For the standard naïve Bayes classifier (Table B.1.4.1), when we vary which terms are involved in the classification, recall is highest when unigrams, title terms and bigrams are considered. Precision and accuracy are both highest when the only terms used are unigrams and title terms. It appears that including bigrams may reduce performance

slightly; the reason why the classifier trained with both title and bigram terms showed the best recall is that, while bigrams from the abstract are not effective features, bigrams from the title, in fact, are effective predictors of the class label. Including late abstract terms does not appear to increase performance here.

When the only-present-terms classifier is employed (Table B.1.4.2), precision and accuracy are highest when the title terms and bigrams are included. Recall is highest when only title terms are included. Again, late abstract terms do not seem to have a beneficial impact on cross-validation performance of classifiers trained on the ***SP-GO*** dataset.

B.1.4.1 Standard Calculation

The effects of modifying the term set on the performance of classifiers trained on the ***SP-GO*** dataset and using the standard calculation. Unigrams are always included in the term set. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	Only unigrams		Title terms included		Bigrams included		Title terms and bigrams included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7257	0.0018	0.7309	0.0012	0.7295	0.0015	0.7333	0.0012
Precision	0.7291	0.0016	0.7365	0.0011	0.7266	0.0012	0.7343	0.0014
Accuracy	0.7284	0.0015	0.7351	0.0009	0.7279	0.0012	0.7343	0.0012

	Title, bigram and 10 late abstract terms included		Title, bigram and 20 late abstract terms included		Title, bigram and 30 late abstract terms included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7317	0.0015	0.7307	0.0011	0.7318	0.0018
Precision	0.7325	0.0015	0.7312	0.0009	0.7307	0.0014
Accuracy	0.7326	0.0011	0.7315	0.0007	0.7314	0.0013

	10 late abstract terms included		20 late abstract terms included		30 late abstract terms included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7238	0.0016	0.7241	0.0020	0.7261	0.0012
Precision	0.7296	0.0015	0.7279	0.0013	0.7284	0.0010
Accuracy	0.7282	0.0014	0.7271	0.0013	0.7281	0.0010

B.1.4.2 Only-Present-Terms Calculation

The effects of modifying the term set on the performance of classifiers trained on the ***SP-GO*** dataset and using the only-present-terms calculation. Unigrams are always included in the term set. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	Only unigrams		Title terms included		Bigrams included		Title terms and bigrams included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7438	0.0011	0.7475	0.0014	0.7352	0.0011	0.7412	0.0013
Precision	0.7202	0.0006	0.7263	0.0012	0.7234	0.0010	0.7296	0.0012
Accuracy	0.7278	0.0005	0.7333	0.0012	0.7275	0.0008	0.7336	0.0008

	Title, bigram and 10 late abstract terms included		Title, bigram and 20 late abstract terms included		Title, bigram and 30 late abstract terms included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7384	0.0014	0.7379	0.0019	0.7373	0.0007
Precision	0.7283	0.0016	0.7272	0.0013	0.7271	0.0012
Accuracy	0.7319	0.0014	0.7310	0.0014	0.7306	0.0009

	10 late abstract terms included		20 late abstract terms included		30 late abstract terms included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7409	0.0013	0.7408	0.0015	0.7406	0.0011
Precision	0.7203	0.0009	0.7196	0.0018	0.7206	0.0012
Accuracy	0.7270	0.0009	0.7265	0.0016	0.7271	0.0010

B.2 Test Results

Classifiers trained on the ***SP-GO*** dataset perform poorly when tested on the ***Curated Journal***, ***Curated Swiss-Prot*** and ***CHAR*** datasets. While varying the feature set does slightly change the performance of the classifier, it does not improve the performance enough for the classifier to correctly label the majority of the articles in any of the datasets. The results over several feature sets are listed in this Section. However, we do not discuss the impact each feature selection method has on the performance, since classifiers trained on the ***SP-GO*** dataset are so clearly ill-suited to label articles in the curated and ***CHAR*** datasets.

B.2.1 Effects of Filtering terms by Document Frequency

B.2.1.1 Standard Calculation

The effects of filtering terms by document frequency on the performance of classifiers trained on the ***SP-GO*** dataset and using the standard calculation. Min indicates the minimum number of articles a term must occur in for it to be included in the feature set. Max indicates the maximum number of articles a term may occur in for it to be included in the feature set. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

		Min 100			Min 30		
		Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
Max 10%	Recall	0.5000	0.2037		0.4063	0.1759	
	Precision	0.4103	0.6000		0.3714	0.5700	
	Accuracy	0.4237	0.3936	0.3020	0.3941	0.3775	0.2980
Max 30%	Recall	0.6250	0.2222		0.4896	0.2037	
	Precision	0.4511	0.6102		0.4017	0.6111	
	Accuracy	0.4631	0.4016	0.3490	0.4138	0.3976	0.3451
Max 60%	Recall	0.5938	0.2222		0.5104	0.2037	
	Precision	0.4419	0.6102		0.4083	0.6168	
	Accuracy	0.4532	0.4016	0.3373	0.4187	0.3996	0.3373

		Min 3		
		Curated Journal	***Curated Swiss-Prot***	***CHAR***
Max 10%	Recall	0.2813	0.1327	
	Precision	0.3462	0.4943	
	Accuracy	0.4089	0.3474	0.2510
Max 30%	Recall	0.2917	0.1420	
	Precision	0.3457	0.5111	
	Accuracy	0.4039	0.3534	0.2667
Max 60%	Recall	0.3021	0.1420	
	Precision	0.3452	0.5111	
	Accuracy	0.3990	0.3534	0.2667

B.2.1.2 Only-Present-Terms Calculation

The effects of filtering terms by document frequency on the performance of classifiers trained on the ***SP-GO*** dataset and using the only-present-terms calculation. Min indicates the minimum number of articles a term must occur in for it to be included in the feature set. Max indicates the maximum number of articles a term may occur in for it to be included in the feature set. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

		Min 100			Min 30		
		Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
Max 10%	Recall	0.5313	0.2068		0.4479	0.1914	
	Precision	0.4215	0.6036		0.3805	0.5849	
	Accuracy	0.4335	0.3956	0.3137	0.3941	0.3855	0.3137
Max 30%	Recall	0.5521	0.2006		0.4896	0.2068	
	Precision	0.4417	0.6132		0.4017	0.6091	
	Accuracy	0.4581	0.3976	0.3177	0.4138	0.3976	0.3137
Max 60%	Recall	0.5625	0.1944		0.3021	0.1512	
	Precision	0.4426	0.6000		0.3537	0.5269	
	Accuracy	0.4581	0.3916	0.3216	0.4089	0.3594	0.2784

		Min 3		
		Curated Journal	***Curated Swiss-Prot***	***CHAR***
Max 10%	Recall	0.3021	0.1512	
	Precision	0.3537	0.5269	
	Accuracy	0.4089	0.3594	0.2784
Max 30%	Recall	0.3021	0.1451	
	Precision	0.3494	0.5054	
	Accuracy	0.4039	0.3514	0.2706
Max 60%	Recall	0.3021	0.1512	
	Precision	0.3537	0.5269	
	Accuracy	0.4089	0.3594	0.2784

B.2.2 Effects of Filtering terms by *Z-Score*

B.2.2.1 Standard Calculation

The effects of filtering terms by *Z-score* on the performance of classifiers trained on the ***SP-GO*** dataset and using the standard calculation. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

		Curated Journal	***Curated Swiss-Prot***	***CHAR***
No *Z-Score* filtering	Recall	0.3021	0.1420	
	Precision	0.3452	0.5111	
	Accuracy	0.3990	0.3534	0.2667
Z = 2	Recall	0.4167	0.1512	
	Precision	0.3846	0.5326	
	Accuracy	0.4089	0.3615	0.2863
Z = 3	Recall	0.4896	0.1667	
	Precision	0.4087	0.5745	
	Accuracy	0.4237	0.3775	0.3216
Z = 4	Recall	0.4896	0.1852	
	Precision	0.3983	0.5882	
	Accuracy	0.4089	0.3855	0.3255
Z = 5	Recall	0.5313	0.2191	
	Precision	0.4180	0.6228	
	Accuracy	0.4286	0.4056	0.3255

B.2.2.2 Only-Present-Terms Calculation

The effects of filtering terms by *Z-score* on the performance of classifiers trained on the ***SP-GO*** dataset and using the only-present-terms calculation. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

		Curated Journal	***Curated Swiss-Prot***	***CHAR***
No *Z-Score* filtering	Recall	0.3021	0.1512	
	Precision	0.3537	0.5269	
	Accuracy	0.4089	0.3594	0.2784
Z = 2	Recall	0.4063	0.1574	
	Precision	0.3861	0.5426	
	Accuracy	0.4138	0.3655	0.2824
Z = 3	Recall	0.4479	0.1512	
	Precision	0.3945	0.5444	
	Accuracy	0.4138	0.3655	0.3098
Z = 4	Recall	0.4375	0.1605	
	Precision	0.3889	0.5714	
	Accuracy	0.4089	0.3755	0.3020
Z = 5	Recall	0.4063	0.1605	
	Precision	0.3824	0.5778	
	Accuracy	0.4089	0.3775	0.2706

B.2.3 Effects of Replacing or Removing Species-Specific terms

B.2.3.1 Standard Calculation

The effects of replacing or removing species-specific terms on the performance of classifiers trained on the ***SP-GO*** dataset and using the standard calculation. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

	Remove species-specific terms			Replace species-specific terms		
	Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
Recall	0.3125	0.1636		0.3125	0.1605	
Precision	0.3529	0.5248		0.3488	0.5253	
Accuracy	0.4039	0.3594	0.2824	0.3990	0.3594	0.2745

B.2.3.2 Only-Present-Terms Calculation

The effects of replacing or removing species-specific terms on the performance of classifiers trained on the ***SP-GO*** dataset and using the only-present-terms calculation. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

	Remove species-specific terms			Replace species-specific terms		
	Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
Recall	0.3646	0.1759		0.3438	0.1667	
Precision	0.3684	0.5429		0.3626	0.5294	
Accuracy	0.4039	0.3675	0.2980	0.4039	0.3615	0.2902

B.2.4 Effects of Modifying the Term set

B.2.4.1 Standard Calculation

The effects of modifying the term set on the performance of classifiers trained on the ***SP-GO*** dataset and using the standard calculation. Unigrams are always included in the term set. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

	Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
	Only unigrams			Title terms included		
Recall	0.3646	0.1574		0.3438	0.1389	
Precision	0.3608	0.5544		0.3511	0.5114	
Accuracy	0.3941	0.3695	0.2863	0.3892	0.3534	0.2471
	Bigrams included			Title terms and bigrams included		
Recall	0.2813	0.1358		0.3021	0.1420	
Precision	0.3177	0.5116		0.3452	0.5111	
Accuracy	0.3744	0.3534	0.2863	0.3990	0.3534	0.2667
	Title, bigram and 10 late abstract terms included			Title, bigram and 20 late abstract terms included		
Recall	0.2917	0.1327		0.2917	0.1389	
Precision	0.3374	0.5000		0.3374	0.5172	
Accuracy	0.3941	0.3494	0.2628	0.3941	0.3554	0.2667
	Title, bigram and 30 late abstract terms included			10 late abstract terms included		
Recall	0.3021	0.1327		0.3646	0.1574	
Precision	0.3412	0.5059		0.3684	0.5368	
Accuracy	0.3941	0.3514	0.2667	0.4039	0.3635	0.3020
	20 late abstract terms included			30 late abstract terms included		
Recall	0.3750	0.1358		0.3854	0.1420	
Precision	0.3711	0.5058		0.3854	0.5287	
Accuracy	0.4039	0.3514	0.2824	0.4187	0.3594	0.2980

B.2.4.2 Only-Present-Terms Calculation

The effects of modifying the term set on the performance of classifiers trained on the ***SP-GO*** dataset and using the only-present-terms calculation. Unigrams are always included in the term set. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

	Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
	Only unigrams			Title terms included		
Recall	0.4375	0.1852		0.3750	0.1605	
Precision	0.4000	0.5769		0.3636	0.5361	
Accuracy	0.4237	0.3815	0.3059	0.3941	0.3635	0.2745
	Bigrams included			Title terms and bigrams included		
Recall	0.3021	0.1358		0.3021	0.1512	
Precision	0.3333	0.5116		0.3537	0.5269	
Accuracy	0.3842	0.3534	0.2902	0.4089	0.3594	0.2784
	Title, bigram and 10 late abstract terms included			Title, bigram and 20 late abstract terms included		
Recall	0.3125	0.1420		0.2917	0.1420	
Precision	0.3529	0.5169		0.3374	0.5227	
Accuracy	0.4039	0.3554	0.2706	0.3941	0.3574	0.2667
	Title, bigram and 30 late abstract terms included			10 late abstract terms included		
Recall	0.3125	0.1358		0.4167	0.1852	
Precision	0.3488	0.5116		0.3922	0.5769	
Accuracy	0.3990	0.3534	0.2667	0.4187	0.3815	0.3059
	20 late abstract terms included			30 late abstract terms included		
Recall	0.3854	0.1698		0.4167	0.1636	
Precision	0.3663	0.5556		0.3960	0.5579	
Accuracy	0.3941	0.3715	0.3059	0.4237	0.3715	0.3020

B.3 Number of Terms in the Feature Sets from the ***SP-GO*** Dataset

The following tables list the number of terms that are included in the feature set under the feature selection methods we evaluated. 'All terms' indicates the total number of unique terms which are used from the articles in the dataset. 'Pos terms' indicates the number of unique terms in the *positive* articles of the dataset, and 'neg terms' indicates the number of unique terms in the *negative* articles.

B.3.1 Effects of filtering by Document Frequency on the Number of Terms

The effects of filtering terms by document frequency on the number of terms in the term set used to represent the articles of the ***SP-GO*** dataset. Min indicates the minimum number of articles a term must occur in for it to be included in the feature set. Max indicates the maximum number of articles a term may occur in for it to be included in the feature set. 'Pos terms' denotes the number of unique terms from the *positive* articles, 'neg terms' denotes the number of unique terms from the *negatives*, and 'all terms' denotes the total number of unique terms.

		Min 100	Min 30	Min 10	Min 3
Max 10%	all terms	1798	5371	15313	60260
	pos terms	1798	5369	15154	53690
	neg terms	1798	5369	15212	53578
Max 20%	all terms	1877	5450	15392	60339
	pos terms	1877	5448	15233	53769
	neg terms	1877	5447	15291	53657
Max 30%	all terms	1905	5478	15420	60367
	pos terms	1905	5476	15261	53797
	neg terms	1905	5475	15319	53685
Max 40%	all terms	1913	5486	15428	60375
	pos terms	1913	5484	15269	53805
	neg terms	1913	5483	15327	53693
Max 60%	all terms	1916	5489	15431	60378
	pos terms	1916	5487	15272	53808
	neg terms	1916	5486	15330	53696

B.3.2 Effects of filtering by *Z-Score* on the Number of Terms

The effects of filtering terms by *Z-score* on the number of terms in the term set used to represent the articles of the ***SP-GO*** dataset. 'Pos terms' denotes the number of unique terms from the *positive* articles, 'neg terms' denotes the number of unique terms from the *negatives*, and 'all terms' denotes the total number of unique terms.

	no Z cutoff	z = 2	z = 3	z = 4	z = 5
all terms	60378	12323	3753	1823	1060
pos terms	53808	10901	3594	1804	1056
neg terms	53696	9500	3592	1806	1055

B.3.3 Effects of Removing Species-Specific terms on the Number of Terms

The effects of replacing or removing species-specific terms on the number of terms in the term set used to represent the articles of the ***SP-GO*** dataset. 'Pos terms' denotes the number of unique terms from the *positive* articles, 'neg terms' denotes the number of unique terms from the *negatives*, and 'all terms' denotes the total number of unique terms.

	Remove	Replace
all terms	55137	57671
pos terms	49687	51865
neg terms	48744	51100

B.3.4 Effects of modifying the Term Set on the Number of Terms

The effects of modifying the term set on the number of terms in the term set used to represent the articles of the ***SP-GO*** dataset. 'Pos terms' denotes the number of unique terms from the *positive* articles, 'neg terms' denotes the number of unique terms from the *negatives*, and 'all terms' denotes the total number of unique terms.

	Only unigrams	Title terms included	Bigrams included	Title terms and bigrams included	Title, bigram and 10 late abstract terms included
all terms	19092	24926	50806	60378	68010
pos terms	17397	22645	45404	53808	60736
neg terms	17284	22412	45368	53696	60527

	Title, bigram and 20 late abstract terms included	Title, bigram and 30 late abstract terms included	10 late abstract terms included	20 late abstract terms included	30 late abstract terms included
all terms	73215	77777	23861	26340	28161
pos terms	65386	69395	21798	24075	25726
neg terms	65169	69215	21626	23812	25422

Appendix C

Cross-Validation and Test Results on the Curated Datasets

C.1 Results for Classifiers Trained on the ***Curated Journal*** Dataset

The following Tables (C.1.1 and C.1.2) show the cross-validation and test results of a classifier trained on the ***Curated Journal*** dataset using the default feature set. Table C.1.3 lists the number of unique terms that are present in the feature set from all the articles in the ***Curated Journal*** dataset ('all terms') as well as the number of unique terms that occur in the *positive* articles ('pos terms') and the *negative* articles of the dataset ('neg terms').

C.1.1 Ten Times 5-Fold Cross-Validation Results

Results for a classifier trained on the ***Curated Journal*** dataset. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation.

	Mean	Standard Deviation
Recall	0.7250	0.0292
Precision	0.7101	0.0171
Accuracy	0.7300	0.0185

C.1.2 Test Results

Results for a classifier trained on the ***Curated Journal*** dataset and tested on the ***Curated Swiss-Prot*** and ***CHAR*** datasets.

	Curated Swiss-Prot	***CHAR***
Recall	0.80864	
Precision	0.72981	
Accuracy	0.68072	0.8037

C.1.3 Number of Terms in the Dataset

The number of unique terms used to represent the articles of the ***Curated Journal*** dataset. 'Pos terms' denotes the number of unique terms from the *positive* articles, 'neg terms' denotes the number of unique terms from the *negatives*, and 'all terms' denotes the total number of unique terms.

All terms	1207
Pos terms	1068
Neg terms	1105

C.2 Results for Classifiers Trained on the ***Curated Swiss-Prot*** Dataset

The following Tables (C.2.1 and C.2.2) show the cross-validation and test results of a classifier trained on the ***Curated Swiss-Prot*** dataset using the default feature set. Table C.2.3 lists the number of unique terms that are present in the feature set from all the articles in the ***Curated Swiss-Prot*** dataset ('all terms') as well as the number of unique terms that occur in the *positive* articles ('pos terms') and the *negative* articles of the dataset.

C.2.1 Ten Times 5-Fold Cross-Validation Results

Results for a classifier trained on the ***Curated Swiss-Prot*** dataset. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation.

	Mean	Standard Deviation
Recall	0.8941	0.0077
Precision	0.8417	0.0053
Accuracy	0.8217	0.0037

C.2.2 Test Results

Results for a classifier trained on the ***Curated Swiss-Prot*** dataset and tested on the ***Curated Journal*** and ***CHAR*** datasets.

	Curated Journal	***CHAR***
Recall	0.9375	
Precision	0.5233	
Accuracy	0.5665	0.8778

C.2.3 Number of Terms in the Dataset

The number of unique terms used to represent the articles of the ***Curated Swiss-Prot*** dataset. 'Pos terms' denotes the number of unique terms from the *positive* articles, 'neg terms' denotes the number of unique terms from the *negatives*, and 'all terms' denotes the total number of unique terms.

All terms	2655
Pos terms	2518
Neg terms	1926

Appendix D

Full Results for Classifiers trained on the ***SPonly*** Dataset

D.1 Cross-Validation results

D.1.1 Effects of Filtering terms by Document Frequency

When varying the document frequency cutoffs used with the standard naïve Bayes classifier, as shown in Table D.1.1.1, performance improved as more terms were included in the classification. It appears that, regardless of how rare or frequent they are, additional features tend to improve the cross-validation performance of the standard classifier trained on the ***SPonly*** dataset.

For the only-present-terms classifier (seen in Table D.1.1.2), when we adjust the document frequency cutoffs, precision and accuracy are highest when many terms are included in the feature set. However, recall is highest when few terms are involved, especially when rare terms are excluded. An increase in recall coupled with a decrease in precision and accuracy indicates that the classifier is labeling many articles as *relevant* regardless of their actual relevance. Very few terms (less than 350) are involved in classification when we exclude terms that do not occur in more than 100 articles. Since only terms that are present in the articles are being considered by this classifier, the terms in the feature set must occur more often in the *positive* training articles than in the *negatives* of the ***SPonly*** dataset.

D.1.1.1 Standard Calculation

The effects of filtering terms by document frequency on the performance of classifiers trained on the ***SPonly*** dataset and using the standard calculation. Min indicates the minimum number of articles a term must occur in for it to be included in the feature set. Max indicates the maximum number of articles a term may occur in for it to be included in the feature set. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

		Max 10%		Max 20%		Max 30%	
		Mean	Stdev	Mean	Stdev	Mean	Stdev
Min 100	Recall	0.6429	0.0047	0.6495	0.0036	0.6620	0.0040
	Precision	0.7057	0.0029	0.7064	0.0028	0.7065	0.0036
	Accuracy	0.6871	0.0025	0.6896	0.0023	0.6933	0.0034
Min 30	Recall	0.6806	0.0036	0.6853	0.0035	0.6870	0.0029
	Precision	0.7216	0.0018	0.7251	0.0023	0.7225	0.0041
	Accuracy	0.7088	0.0012	0.7125	0.0024	0.7114	0.0030
Min 10	Recall	0.6979	0.0032	0.6978	0.0029	0.6993	0.0031
	Precision	0.7207	0.0027	0.7183	0.0047	0.7200	0.0031
	Accuracy	0.7135	0.0024	0.7118	0.0036	0.7135	0.0024
Min 3	Recall	0.7110	0.0046	0.7102	0.0050	0.7132	0.0039
	Precision	0.7176	0.0061	0.7178	0.0040	0.7184	0.0041
	Accuracy	0.7154	0.0054	0.7153	0.0031	0.7166	0.0034

		Max 40%		Max 50%		Max 60%	
		Mean	Stdev	Mean	Stdev	Mean	Stdev
Min 100	Recall	0.6633	0.0032	0.6653	0.0026	0.6648	0.0024
	Precision	0.7049	0.0034	0.7057	0.0019	0.7053	0.0028
	Accuracy	0.6926	0.0029	0.6937	0.0013	0.6933	0.0019
Min 30	Recall	0.6890	0.0032	0.6905	0.0040	0.6908	0.0035
	Precision	0.7214	0.0025	0.7220	0.0016	0.7211	0.0031
	Accuracy	0.7113	0.0024	0.7121	0.0018	0.7116	0.0023
Min 10	Recall	0.7012	0.0050	0.7011	0.0028	0.7011	0.0035
	Precision	0.7200	0.0044	0.7184	0.0033	0.7192	0.0036
	Accuracy	0.7140	0.0043	0.7130	0.0027	0.7135	0.0030
Min 3	Recall	0.7117	0.0042	0.7117	0.0048	0.7112	0.0066
	Precision	0.7164	0.0034	0.7182	0.0045	0.7161	0.0045
	Accuracy	0.7148	0.0032	0.7160	0.0038	0.7144	0.0036

D.1.1.2 Only-Present-Terms Calculation

The effects of filtering terms by document frequency on the performance of classifiers trained on the ***SPonly*** dataset and using the only-present-terms calculation. Min indicates the minimum number of articles a term must occur in for it to be included in the feature set. Max indicates the maximum number of articles a term may occur in for it to be included in the feature set. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

		Max 10%		Max 20%		Max 30%	
		Mean	Stdev	Mean	Stdev	Mean	Stdev
Min 100	Recall	0.8433	0.0054	0.8037	0.0038	0.8339	0.0032
	Precision	0.6235	0.0020	0.6426	0.0025	0.6355	0.0021
	Accuracy	0.6668	0.0024	0.6781	0.0026	0.6776	0.0024
Min 30	Recall	0.7994	0.0022	0.7888	0.0033	0.7992	0.0037
	Precision	0.6598	0.0026	0.6665	0.0026	0.6609	0.0023
	Accuracy	0.6934	0.0025	0.6969	0.0024	0.6943	0.0024
Min 10	Recall	0.7813	0.0035	0.7737	0.0039	0.7826	0.0037
	Precision	0.6717	0.0020	0.6751	0.0022	0.6731	0.0013
	Accuracy	0.6996	0.0020	0.7005	0.0026	0.7010	0.0013
Min 3	Recall	0.7648	0.0062	0.7612	0.0038	0.7681	0.0036
	Precision	0.6943	0.0036	0.6911	0.0025	0.6929	0.0025
	Accuracy	0.7138	0.0042	0.7102	0.0026	0.7136	0.0026

		Max 40%		Max 50%		Max 60%	
		Mean	Stdev	Mean	Stdev	Mean	Stdev
Min 100	Recall	0.8467	0.0035	0.8407	0.0034	0.8458	0.0034
	Precision	0.6301	0.0023	0.6337	0.0025	0.6312	0.0021
	Accuracy	0.6746	0.0027	0.6771	0.0032	0.6756	0.0025
Min 30	Recall	0.8105	0.0047	0.8058	0.0029	0.8095	0.0027
	Precision	0.6582	0.0020	0.6603	0.0029	0.6589	0.0025
	Accuracy	0.6946	0.0027	0.6954	0.0031	0.6950	0.0028
Min 10	Recall	0.7889	0.0034	0.7875	0.0027	0.7897	0.0040
	Precision	0.6708	0.0031	0.6718	0.0025	0.6729	0.0020
	Accuracy	0.7007	0.0032	0.7012	0.0026	0.7027	0.0022
Min 3	Recall	0.7704	0.0039	0.7683	0.0037	0.7691	0.0050
	Precision	0.6912	0.0017	0.6910	0.0034	0.6905	0.0060
	Accuracy	0.7129	0.0020	0.7121	0.0029	0.7119	0.0056

D.1.2 Effects of Filtering terms by *Z-Score*

For the standard naïve Bayes classifier, increasing the *Z-score* threshold resulted in increased precision but reduced recall (Table D.1.2.1).

In contrast, for the only-present-terms classifier, we see the highest recall (89.32%), but also the lowest precision results (60.39%) over all the classifiers at the

highest *Z-score* threshold, $Z = 5$ (Table D.1.2.2). Accuracy was highest for both classifiers at lower *Z-score* thresholds. Figure D.1 displays these trends. Again, we observe that when few terms are included in the feature set (176 terms at $Z = 4$, 86 terms at $Z = 5$), the only-present-terms classifier tends to label articles as *positive*, resulting in increased recall but decreased precision and accuracy.

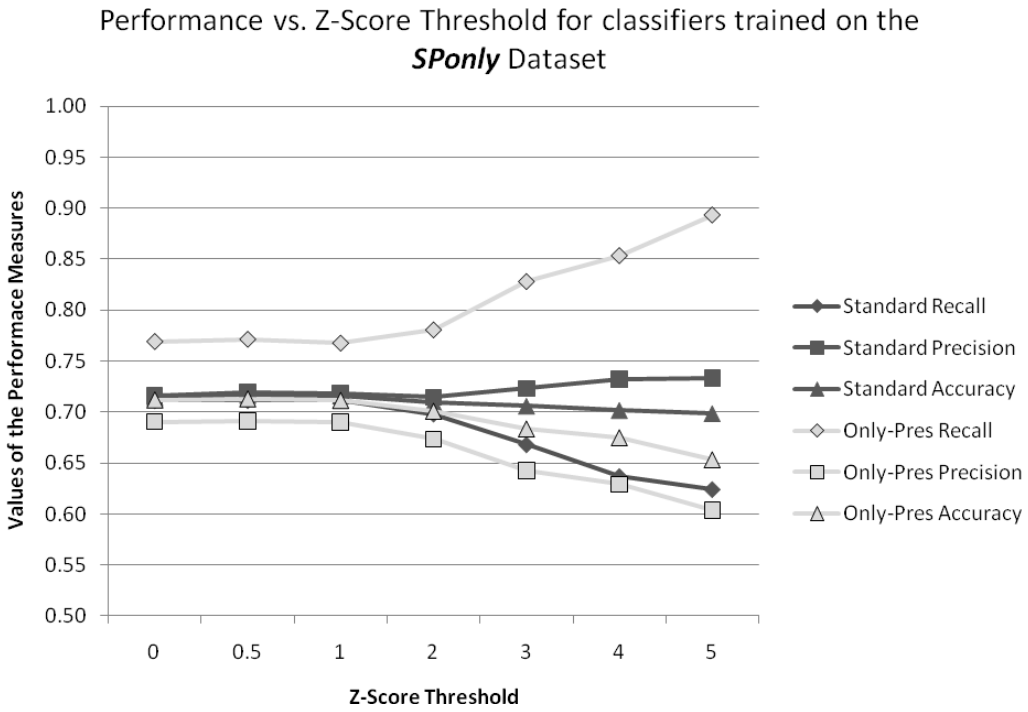


Figure D.1: Classification results at several *Z-score* thresholds for classifiers trained and tested on the ***SPonly*** dataset using cross-validation. Only-Pres denotes the only-present-terms classifier. All terms are included when the *Z-Score* threshold is 0.

D.1.2.1 Standard Calculation

The effects of filtering terms by *Z-score* on the performance of classifiers trained on the ***SPonly*** dataset and using the standard calculation. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	Z = 0.5		Z = 1		Z = 2	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7112	0.0075	0.7119	0.0073	0.6981	0.0061
Precision	0.7193	0.0036	0.7184	0.0031	0.7145	0.0033
Accuracy	0.7166	0.0039	0.7162	0.0031	0.7094	0.0036

	Z = 3		Z = 4		Z = 5	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.6680	0.0063	0.6369	0.0057	0.6242	0.0051
Precision	0.7231	0.0022	0.7320	0.0030	0.7334	0.0032
Accuracy	0.7059	0.0024	0.7017	0.0023	0.6985	0.0028

D.1.2.2 Only-Present-Terms Calculation

The effects of filtering terms by *Z-score* on the performance of classifiers trained on the ***SPonly*** dataset and using the only-present-terms calculation. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	Z = 0.5		Z = 1		Z = 2	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7713	0.0053	0.7675	0.0048	0.7805	0.0065
Precision	0.6910	0.0042	0.6903	0.0018	0.6734	0.0024
Accuracy	0.7129	0.0040	0.7114	0.0021	0.7008	0.0028

	Z = 3		Z = 4		Z = 5	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.8279	0.0040	0.8533	0.0052	0.8932	0.0048
Precision	0.6427	0.0025	0.6292	0.0022	0.6039	0.0039
Accuracy	0.6835	0.0029	0.6750	0.0027	0.6534	0.0051

D.1.3 Effects of Removing or Replacing Species-Specific terms

Removing or replacing species-specific terms from either classifier decreases performance slightly (Table D.1.3.1 and D.1.3.2). A decrease in performance, resulting from removing these terms, is expected as these terms can be powerful predictors of class

within a dataset. As discussed in Section 4.2.3, our reason for removing these terms is to avoid bias for articles discussing certain species in future classifications.

D.1.3.1 Standard Calculation

The effects of removing or replacing species-specific terms on the performance of classifiers trained on the ***SPonly*** dataset and using the standard calculation. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	Replace species-specific terms		Remove species-specific terms	
	Mean	Stdev	Mean	Stdev
Recall	0.7089	0.0030	0.7087	0.0046
Precision	0.7170	0.0025	0.7177	0.0050
Accuracy	0.7143	0.0020	0.7148	0.0037

D.1.3.2 Only-Present-Terms Calculation

The effects of removing or replacing species-specific terms on the performance of classifiers trained on the ***SPonly*** dataset and using the only-present-terms calculation. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	Replace species-specific terms		Remove species-specific terms	
	Mean	Stdev	Mean	Stdev
Recall	0.7678	0.0043	0.7629	0.0049
Precision	0.6893	0.0043	0.6905	0.0036
Accuracy	0.7107	0.0040	0.7103	0.0036

D.1.4 Effects of Modifying the Term Set

For the standard naïve Bayes classifier, the inclusion of title terms and bigrams in the classification appears to improve cross-validation performance when compared to classifiers trained without these terms (Table D.1.4.1). The best cross-validation performance of any ***SPonly*** trained classifier using the standard naïve Bayes calculation occurs when title terms, bigrams, and the last 10 abstract terms are used for classification.

When varying which terms are included while training the *only-present-terms* classifier (Table D.1.4.2) the addition of title and bigram terms both increase precision and accuracy slightly (at the cost of recall). Accuracy is highest (0.7134) when title terms, bigrams and the last 20 abstract terms are considered by the classifier.

D.1.4.1 Standard Calculation

The effects of modifying the term set on the performance of classifiers trained on the ***SPonly*** dataset and using the standard calculation. Unigrams are always included in the term set. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	Only unigrams		Title terms included		Bigrams included		Title terms and bigrams included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7022	0.0052	0.7086	0.0039	0.7105	0.0047	0.7121	0.0052
Precision	0.7091	0.0031	0.7127	0.0048	0.7142	0.0053	0.7155	0.0058
Accuracy	0.7069	0.0031	0.7113	0.0036	0.7129	0.0043	0.7143	0.0053

	Title, bigram and 5 late abstract terms included		Title, bigram and 10 late abstract terms included		Title, bigram and 15 late abstract terms included		Title, bigram and 20 late abstract terms included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7088	0.0036	0.7125	0.0044	0.7100	0.0049	0.7101	0.0061
Precision	0.7137	0.0042	0.7181	0.0031	0.7157	0.0043	0.7185	0.0045
Accuracy	0.7120	0.0027	0.7161	0.0023	0.7138	0.0040	0.7158	0.0043

	Title, bigram and 30 late abstract terms included		5 late abstract terms included		10 late abstract terms included		15 late abstract terms included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7098	0.0061	0.6988	0.0038	0.7015	0.0048	0.7041	0.0042
Precision	0.7140	0.0034	0.7049	0.0033	0.7036	0.0033	0.7060	0.0042
Accuracy	0.7125	0.0033	0.7029	0.0030	0.7028	0.0029	0.7052	0.0035

	20 late abstract terms included		30 late abstract terms included	
	Mean	Stdev	Mean	Stdev
Recall	0.6990	0.0043	0.6967	0.0046
Precision	0.7106	0.0047	0.7083	0.0032
Accuracy	0.7069	0.0039	0.7047	0.0027

D.1.4.2 Only-Present-Terms Calculation

The effects of modifying the term set on the performance of classifiers trained on the ***SPonly*** dataset and using the only-present-terms calculation. Unigrams are always included in the term set. The mean recall, precision and accuracy over ten 5-fold cross-validations are reported, along with the standard deviation (Stdev).

	Only unigrams		Title terms included		Bigrams included		Title terms and bigrams included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7802	0.0074	0.7730	0.0067	0.7753	0.0068	0.7686	0.0053
Precision	0.6702	0.0032	0.6798	0.0038	0.6824	0.0030	0.6904	0.0037
Accuracy	0.6979	0.0038	0.7043	0.0048	0.7070	0.0039	0.7118	0.0037

	Title, bigram and 5 late abstract terms included		Title, bigram and 10 late abstract terms included		Title, bigram and 15 late abstract terms included		Title, bigram and 20 late abstract terms included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7656	0.0027	0.7625	0.0060	0.7592	0.0046	0.7626	0.0060
Precision	0.6905	0.0040	0.6916	0.0036	0.6902	0.0021	0.6946	0.0038
Accuracy	0.7110	0.0036	0.7110	0.0040	0.7090	0.0025	0.7134	0.0041

	Title, bigram and 30 late abstract terms included		5 late abstract terms included		10 late abstract terms included		15 late abstract terms included	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Recall	0.7540	0.0052	0.7693	0.0062	0.7729	0.0038	0.7664	0.0044
Precision	0.6958	0.0024	0.6652	0.0044	0.6743	0.0039	0.6755	0.0033
Accuracy	0.7120	0.0019	0.6908	0.0050	0.6995	0.0034	0.6989	0.0034

	20 late abstract terms included		30 late abstract terms included	
	Mean	Stdev	Mean	Stdev
Recall	0.7612	0.0044	0.7579	0.0045
Precision	0.6795	0.0028	0.6775	0.0026
Accuracy	0.7009	0.0026	0.6984	0.0031

D.2 Test Results

D.2.1 Effects of Filtering terms by Document Frequency

For the standard naïve Bayes classifier, when we varied the minimum and maximum document frequency cutoffs, performance was best when more terms were included, as can be seen in Table D.2.1.1.

However, as shown in Table D.2.1.2, when examining the impact of document frequency filtering with the only-present-terms classifier, no clear trend emerges.

Performance over the ***Curated Journal*** dataset appears best when terms that occur in fewer than 30 articles and more than 10% of the articles were excluded. Performance over the ***Curated Swiss-Prot*** dataset is best when terms that occur in fewer than 30 articles and more than 30% of the articles were excluded. Performance over the ***CHAR*** dataset is best when terms that occur in fewer than 100 articles or more than 60% of the articles were excluded.

D.2.1.1 Standard Calculation

The effects of filtering terms by document frequency on the performance of classifiers trained on the ***SPonly*** dataset and using the standard calculation. Min indicates the minimum number of articles a term must occur in for it to be included in the feature set. Max indicates the maximum number of articles a term may occur in for it to be included in the feature set. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

		Min 100			Min 30		
		Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
Max 10%	Recall	0.5729	0.5833		0.6354	0.6605	
	Precision	0.4701	0.8514		0.4919	0.8699	
	Accuracy	0.4926	0.6627	0.5725	0.5172	0.7149	0.6706
Max 30%	Recall	0.5938	0.6327		0.6042	0.6914	
	Precision	0.4711	0.8577		0.4640	0.8615	
	Accuracy	0.4926	0.6928	0.6118	0.4828	0.7269	0.6784
Max 60%	Recall	0.5729	0.6204		0.6250	0.6790	
	Precision	0.4783	0.8739		0.4878	0.8661	
	Accuracy	0.5025	0.6948	0.6000	0.5123	0.7229	0.6784

		Min 3		
		Curated Journal	***Curated Swiss-Prot***	***CHAR***
Max 10%	Recall	0.7292	0.7099	
	Precision	0.5185	0.8394	
	Accuracy	0.5517	0.7229	0.7216
Max 30%	Recall	0.7396	0.7160	
	Precision	0.5108	0.8406	
	Accuracy	0.5419	0.7269	0.7294
Max 60%	Recall	0.7396	0.7037	
	Precision	0.5221	0.8382	
	Accuracy	0.5567	0.7189	0.7098

D.2.1.2 Only-Present-Terms Calculation

The effects of filtering terms by document frequency on the performance of classifiers trained on the ***SPonly*** dataset and using the only-present-terms calculation. Min indicates the minimum number of articles a term must occur in for it to be included in the feature set. Max indicates the maximum number of articles a term may occur in for it to be included in the feature set. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

		Min 100			Min 30		
		Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
Max 10%	Recall	0.8958	0.8302		0.8750	0.8086	
	Precision	0.4943	0.7730		0.4970	0.7939	
	Accuracy	0.5172	0.7309	0.8863	0.5222	0.7390	0.8745
Max 30%	Recall	0.8021	0.8519		0.8438	0.8395	
	Precision	0.4611	0.7797		0.4850	0.8047	
	Accuracy	0.4631	0.7470	0.8941	0.5025	0.7631	0.8706
Max 60%	Recall	0.8333	0.8827		0.8646	0.8457	
	Precision	0.4706	0.7730		0.4940	0.7965	
	Accuracy	0.4778	0.7550	0.9098	0.5172	0.7590	0.8784

		Min 3		
		Curated Journal	***Curated Swiss-Prot***	***CHAR***
Max 10%	Recall	0.7917	0.7809	
	Precision	0.4903	0.8241	
	Accuracy	0.5123	0.7490	0.7961
Max 30%	Recall	0.7917	0.7870	
	Precision	0.4872	0.8173	
	Accuracy	0.5074	0.7470	0.7961
Max 60%	Recall	0.7917	0.7932	
	Precision	0.4903	0.8159	
	Accuracy	0.5123	0.7490	0.8000

D.2.2 Effects of Filtering terms by *Z-Score*

For both the standard and the only-present-terms classifiers, performance tends to decrease as the *Z-score* threshold is increased, which is expected because very few terms are involved in the classification at the higher *Z-score* thresholds (176 terms at $Z = 4$, and 86 terms at $Z = 5$). The exception was the only-present-term classifier performance over the ***CHAR*** validation set, which showed the highest accuracy when the *Z-score* threshold was 3 or 5. Since the CHAR set only contains *positive* articles, the increased accuracy of

the only-present-term classifier is likely due to its tendency to classify articles as *positive* when few terms are included in the feature set, which we have discussed in Section 5.5.2. Figure D.2 shows the accuracy of the standard and of the only-present terms classifiers trained on the ***SPonly*** dataset, and tested over the other datasets. These results are shown in Table D.2.2.1 and D.2.2.2. The best classifier under any combination of feature parameters tested over the ***Curated Journal*** dataset was a standard naïve Bayes classifier trained with a *Z-score* threshold of 0.5.

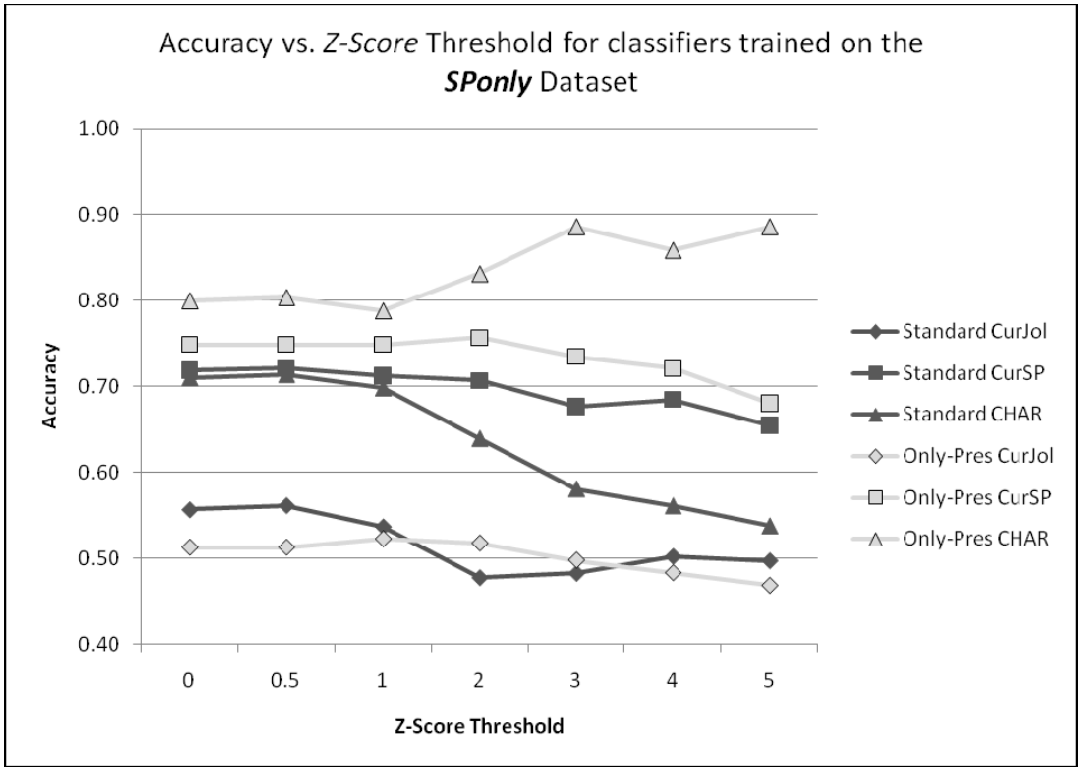


Figure D.2: Accuracy at several *Z-score* thresholds for classifiers trained on the ***SPonly*** dataset and tested on the other datasets. Only-Pres denotes the only-present-terms classifier. All terms are included when the *Z-Score* threshold is 0.

D.2.2.1 Standard Calculation

The effects of filtering terms by *Z-score* on the performance of classifiers trained on the ***SPonly*** dataset and using the standard calculation. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

		Curated Journal	***Curated Swiss-Prot***	***CHAR***
Z = 0.5	Recall	0.7396	0.7068	
	Precision	0.5259	0.8388	
	Accuracy	0.5616	0.7209	0.7137
Z = 1	Recall	0.7292	0.6914	
	Precision	0.5072	0.8390	
	Accuracy	0.5369	0.7129	0.6980
Z = 2	Recall	0.6146	0.6698	
	Precision	0.4609	0.8477	
	Accuracy	0.4778	0.7068	0.6392
Z = 3	Recall	0.5521	0.6080	
	Precision	0.4609	0.8528	
	Accuracy	0.4828	0.6767	0.5804
Z = 4	Recall	0.5208	0.5957	
	Precision	0.4762	0.8813	
	Accuracy	0.5025	0.6847	0.5608
Z = 5	Recall	0.4896	0.5617	
	Precision	0.4700	0.8585	
	Accuracy	0.4975	0.6546	0.5373

D.2.2.2 Only-Present-Terms Calculation

The effects of filtering terms by *Z-score* on the performance of classifiers trained on the ***SPonly*** dataset and using the only-present-terms calculation. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

		Curated Journal	***Curated Swiss-Prot***	***CHAR***
Z = 0.5	Recall	0.8021	0.7932	
	Precision	0.4904	0.8159	
	Accuracy	0.5123	0.7490	0.8039
Z = 1	Recall	0.8021	0.7840	
	Precision	0.4968	0.8220	
	Accuracy	0.5222	0.7490	0.7882
Z = 2	Recall	0.8333	0.8210	
	Precision	0.4938	0.8085	
	Accuracy	0.5172	0.7570	0.8314
Z = 3	Recall	0.8542	0.8519	
	Precision	0.4824	0.7667	
	Accuracy	0.4975	0.7349	0.8863
Z = 4	Recall	0.8646	0.8488	
	Precision	0.4743	0.7534	
	Accuracy	0.4828	0.7209	0.8588
Z = 5	Recall	0.8750	0.8457	
	Precision	0.4667	0.7154	
	Accuracy	0.4680	0.6807	0.8863

D.2.3 Effect of Removing or Replacing Species-Specific terms

For both the standard and only-present-terms classifiers, performance tended to be slightly lower when species-specific terms were replaced or removed (Tables D.2.3.1 and D.2.3.2). As discussed above, and in Section 4.2.3, this reduction in performance is expected.

D.2.3.1 Standard Calculation

The effects of replacing or removing species-specific terms on the performance of classifiers trained on the ***SPonly*** dataset and using the standard calculation. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

	Remove species-specific terms			Replace species-specific terms		
	Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
Recall	0.6563	0.6636		0.6667	0.6821	
Precision	0.5000	0.8498		0.5039	0.8435	
Accuracy	0.5271	0.7048	0.6549	0.5320	0.6987	0.6549

D.2.3.2 Only-Present-Terms Calculation

The effects of replacing or removing species-specific terms on the performance of classifiers trained on the ***SPonly*** dataset and using the only-present-terms calculation. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

	Remove species-specific terms			Replace species-specific terms		
	Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
Recall	0.7604	0.7623		0.7604	0.7778	
Precision	0.4932	0.8179		0.4899	0.8182	
Accuracy	0.5172	0.7349	0.7725	0.5123	0.7430	0.7843

D.2.4 Effects of Modifying the Term set

For the standard naïve Bayes classifier, adding title terms and late abstract terms tended to improved classifier performance over the datasets (Table D.2.4.1). Interestingly, the classifier that used bigrams in its feature set showed worse performance than the

classifier that used only unigrams. The best classifier on the ***Curated Swiss-Prot*** dataset used unigrams and title terms as features.

When varying the feature set while training the *only-present-terms* classifier, excluding the title and bigram terms while including the last 5 or 20 abstract terms, appears to yield the highest performance when applied to all the other datasets (Table D.2.4.2).

D.2.4.1 Standard Calculation

The effects of modifying the term set on the performance of classifiers trained on the ***SPonly*** dataset and using the standard calculation. Unigrams are always included in the term set. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

	Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
	Only unigrams			Title terms included		
Recall	0.6667	0.6914		0.7188	0.7099	
Precision	0.4885	0.8550		0.5111	0.8679	
Accuracy	0.5123	0.7229	0.6784	0.5419	0.7410	0.6980
	Bigrams included			Title terms and bigrams included		
Recall	0.6875	0.6759		0.7396	0.7037	
Precision	0.4889	0.8391		0.5221	0.8382	
Accuracy	0.5123	0.7048	0.7059	0.5567	0.7189	0.7098
	Title, bigram and 5 late abstract terms included			Title, bigram and 10 late abstract terms included		
Recall	0.7292	0.7099		0.7188	0.7160	
Precision	0.5147	0.8456		0.5036	0.8467	
Accuracy	0.5468	0.7269	0.7098	0.5320	0.7309	0.7176
	Title, bigram and 15 late abstract terms included			Title, bigram and 20 late abstract terms included		
Recall	0.7396	0.6975		0.7396	0.6944	
Precision	0.5071	0.8433		0.5000	0.8333	
Accuracy	0.5369	0.7189	0.7216	0.5271	0.7108	0.7098
	Title, bigram and 30 late abstract terms included			5 late abstract terms included		
Recall	0.7396	0.7160		0.6667	0.7130	
Precision	0.5000	0.8436		0.4885	0.8524	
Accuracy	0.5271	0.7289	0.7216	0.5123	0.7329	0.6824
	10 late abstract terms included			15 late abstract terms included		
Recall	0.6667	0.7284		0.6875	0.7068	
Precision	0.4776	0.8489		0.4853	0.8609	
Accuracy	0.4975	0.7390	0.6824	0.5074	0.7349	0.6784
	20 late abstract terms included			30 late abstract terms included		
Recall	0.6979	0.7160		0.6875	0.7284	
Precision	0.4926	0.8467		0.4925	0.8310	
Accuracy	0.5172	0.7309	0.6980	0.5172	0.7269	0.7020

D.2.4.2 Only-Present-Terms Calculation

The effects of modifying the term set on the performance of classifiers trained on the ***SPonly*** dataset and using the only-present-terms calculation. Unigrams are always included in the term set. The recall, precision and accuracy over the curated and ***CHAR*** datasets are reported here.

	Curated Journal	***Curated Swiss-Prot***	***CHAR***	***Curated Journal***	***Curated Swiss-Prot***	***CHAR***
	Only unigrams			Title terms included		
Recall	0.8542	0.8179		0.8333	0.8179	
Precision	0.4970	0.8006		0.4969	0.8129	
Accuracy	0.5222	0.7490	0.8431	0.5222	0.7590	0.8157
	Bigrams included			Title terms and bigrams included		
Recall	0.8125	0.7963		0.7917	0.7932	
Precision	0.4845	0.8217		0.4903	0.8159	
Accuracy	0.5025	0.7550	0.8157	0.5123	0.7490	0.8000
	Title, bigram and 5 late abstract terms included			Title, bigram and 10 late abstract terms included		
Recall	0.7813	0.8056		0.7917	0.7901	
Precision	0.4934	0.8233		0.4903	0.8285	
Accuracy	0.5172	0.7610	0.7922	0.5123	0.7570	0.8039
	Title, bigram and 15 late abstract terms included			Title, bigram and 20 late abstract terms included		
Recall	0.7917	0.7778		0.8021	0.7870	
Precision	0.4903	0.8235		0.4936	0.8226	
Accuracy	0.5123	0.7470	0.8000	0.5172	0.7510	0.7922
	Title, bigram and 30 late abstract terms included			5 late abstract terms included		
Recall	0.7813	0.7778		0.8229	0.8241	
Precision	0.4902	0.8317		0.4907	0.8091	
Accuracy	0.5123	0.7530	0.7961	0.5123	0.7590	0.8431
	10 late abstract terms included			15 late abstract terms included		
Recall	0.8125	0.8395		0.8229	0.8117	
Precision	0.4968	0.8047		0.4969	0.8092	
Accuracy	0.5222	0.7631	0.8353	0.5222	0.7530	0.8196
	20 late abstract terms included			30 late abstract terms included		
Recall	0.8229	0.8179		0.8125	0.7932	
Precision	0.5000	0.8204		0.4875	0.8107	
Accuracy	0.5271	0.7651	0.8392	0.5074	0.7450	0.8157

D.3 Number of Terms in the Feature Sets from the ***SPonly*** Dataset

The following tables list the number of terms that are included in the feature set under the feature selection methods we evaluated. 'All terms' indicates the total number of unique terms which are used from the articles in the dataset. 'Pos terms' indicates the number of

unique terms in the *positive* articles of the dataset, and 'neg terms' indicates the number of unique terms in the *negative* articles.

D.3.1 Effect of filtering by Document Frequency on the Number of Terms

The effects of filtering terms by document frequency on the number of terms in the term set used to represent the articles of the ***SPonly*** dataset. Min indicates the minimum number of articles a term must occur in for it to be included in the feature set. Max indicates the maximum number of articles a term may occur in for it to be included in the feature set. 'Pos terms' denotes the number of unique terms from the *positive* articles, 'neg terms' denotes the number of unique terms from the *negatives*, and 'all terms' denotes the total number of unique terms.

		Min 100	Min 30	Min 10	Min 3
Max 10%	all terms	184	936	2927	11384
	pos terms	184	936	2920	10254
	neg terms	184	936	2908	10060
Max 20%	all terms	276	1028	3019	11476
	pos terms	276	1028	3012	10346
	neg terms	276	1028	3000	10152
Max 30%	all terms	298	1050	3041	11498
	pos terms	298	1050	3034	10368
	neg terms	298	1050	3022	10174
Max 40%	all terms	308	1060	3051	11508
	pos terms	308	1060	3044	10378
	neg terms	308	1060	3032	10184
Max 50%	all terms	313	1065	3056	11513
	pos terms	313	1065	3049	10383
	neg terms	313	1065	3037	10189
Max 60%	all terms	314	1066	3057	11514
	pos terms	314	1066	3050	10384
	neg terms	314	1066	3038	10190

D.3.2 Effect of filtering by *Z-Score* on the Number of Terms

The effects of filtering terms by *Z-score* on the number of terms in the term set used to represent the articles of the ***SPonly*** dataset. 'Pos terms' denotes the number of unique terms from the *positive* articles, 'neg terms' denotes the number of unique terms from the *negatives*, and 'all terms' denotes the total number of unique terms.

	z = 0.5	z = 1	z = 2	z = 3	z = 4	z = 5
all terms	9168	7093	2218	451	176	86
pos terms	8038	5963	1767	437	175	86
neg terms	7844	5769	1700	426	174	86

D.3.3 Effect of Removing Species-Specific terms on the Number of Terms

The effects of replacing or removing species-specific terms on the number of terms in the term set used to represent the articles of the ***SPonly*** dataset. 'Pos terms' denotes the number of unique terms from the *positive* articles, 'neg terms' denotes the number of unique terms from the *negatives*, and 'all terms' denotes the total number of unique terms.

	Remove	Replace
all terms	10441	10915
pos terms	9450	9863
neg terms	9265	9702

D.3.4 Effect of modifying the Term Set on the Number of Terms

The effects of modifying the term set on the number of terms in the term set used to represent the articles of the ***SPonly*** dataset. 'Pos terms' denotes the number of unique terms from the *positive* articles, 'neg terms' denotes the number of unique terms from the *negatives*, and 'all terms' denotes the total number of unique terms.

	Only unigrams	Title terms included	Bigrams included	Title terms and bigrams included
all terms	5280	6548	9838	11514
pos terms	4889	6032	8900	10384
neg terms	4802	5923	8726	10190

	Title, bigram and 5 late abstract terms included	Title, bigram and 10 late abstract terms included	Title, bigram and 15 late abstract terms included	Title, bigram and 20 late abstract terms included	Title, bigram and 30 late abstract terms included
all terms	12430	13038	13632	14121	14995
pos terms	11207	11777	12296	12751	13540
neg terms	11006	11557	12088	12519	13308

	5 late abstract terms included	10 late abstract terms included	15 late abstract terms included	20 late abstract terms included	30 late abstract terms included
all terms	6082	6570	6973	7281	7786
pos terms	5617	6082	6446	6746	7232
neg terms	5530	5968	6330	6616	7101

Appendix E

Top terms by *Z-Score* from the ***SP-GO*** and ***SPonly*** Datasets

This appendix lists the terms with the highest *Z-scores* from the ***SP-GO*** and ***SPonly*** datasets. A term's *Z-score* indicates the statistical significance of the difference between the probability of the term to occur in the *positive* articles and the probability of the term to occur in the *negative* articles of a dataset, as discussed in Section 4.2.2. In these tables, 'pos count' indicates the number of *positive* articles that contain the term in the dataset, while 'neg count' indicates the number of *negative* articles that contain the term.

E.1 Top 50 terms by *Z-score* from the ***SP-GO*** *Dataset.*

The top 50 terms by *Z-score* from the ***SP-GO*** dataset. 'Pos count' indicates the number of *positive* articles that contain the term in the dataset, while 'neg count' indicates the number of *negative* articles that contain the term. Terms are stemmed, and !T indicates a title term.

Term	*Z-score*	pos count	neg count
Sequenc	-38.5884	3106	5894
Interact	31.63643	3244	1398
Clone	-31.0046	2028	4101
gene!T	-30.1797	1274	3071
Gene	-27.369	4600	6613
active	26.4605	5420	3642
fission yeast	26.43377	1011	148
Fission	26.32767	1024	157
Cdna	-25.9546	1536	3128
Pomb	25.42702	1048	192
Schizosaccharomyc	25.1517	1013	181
schizosaccharomyc pomb	25.13144	1012	181
Encod	-24.6539	2811	4555
Genom	-23.9271	818	2024
sequenc!T	-23.6422	227	1058
amino acid	-23.1605	2128	3663
interact!T	22.36503	1060	281
Complex	22.32567	2521	1313
Yeast	22.2993	2349	1179
Code	-21.768	471	1379

Cell	21.76008	6090	4656
Amino	-21.6784	2346	3814
fission yeast!T	21.21809	656	93
fission!T	21.20638	666	98
Acid	-20.9699	2805	4284
Bind	20.60566	3766	2473
Isol	-20.1602	1785	3047
require	19.99674	2443	1367
pomb!T	19.89283	579	82
Kb	-19.7992	216	856
Inhibit	19.64633	1943	987
clone!T	-19.6038	718	1631
Kinas	19.46086	1703	813
genom!T	-19.3794	63	535
Frame	-19.2229	458	1228
schizosaccharomyc pomb!T	19.2123	542	77
schizosaccharomyc!T	19.18305	543	78
Analysi	-19.0861	2339	3625
Read	-19.0387	449	1205
Mediat	18.79175	2066	1122
Speci	-18.7487	553	1347
read frame	-18.712	438	1172
open read	-18.6317	436	1165
Ident	-18.3151	1141	2127
express!T	-18.2643	885	1787
nucleotid sequenc	-18.20625	176	715
Deduc	-18.0735	362	1023
Associ	17.77079	2699	1698
Bp	-17.7681	224	782
Open	-17.5193	523	1241

E.2 Top 50 terms by *Z-score* from the ***SPonly*** Dataset

The top 50 terms by *Z-score* from the ***SPonly*** dataset. 'Pos count' indicates the number of *positive* articles that contain the term in the dataset, while 'neg count' indicates the number of *negative* articles that contain the term. Terms are stemmed, and !T indicates a title term.

term	*Z-score*	pos count	neg count
purify	15.34456	483	143
enzym	14.35726	542	204
activ	13.7981	868	497
coli	12.39215	407	145
escherichia	12.23957	390	136
escherichia coli	12.21859	388	135
substrat	11.43358	284	80
character!T	11.25385	465	209
catalyz	10.41105	194	41
purif	9.693084	190	47
purif!T	9.538411	144	24
character	9.44781	715	465
homogen	9.193977	153	32

ph	8.923597	166	42
genom!T	-8.52801	9	94
properti	8.268589	222	85
microm	7.770083	103	19
molecular mass	7.725183	208	83
inhibit	7.723645	241	106
M	7.683415	226	96
mass	7.623423	261	122
E	7.581428	256	119
mous	-7.29262	131	266
gel	7.197081	138	44
chromatographi	7.193491	133	41
phosphat	7.191009	86	15
degre c	7.166119	112	29
recombin	7.15365	255	125
km	7.130092	81	13
pattern	-7.02421	78	187
gene!T	-6.95649	313	480
valu	6.953675	124	38
escherichia coli!T	6.857519	100	25
region	-6.7497	301	461
escherichia!T	6.740396	100	26
coli!T	6.527061	102	29
purifi enzym	6.519788	64	9
nativ	6.518084	112	35
k	6.506638	154	62
residu	6.391704	337	203
optimum	6.358972	64	10
kinet	6.222005	94	27
evolut	-6.20322	26	92
overexpress	5.976478	162	74
kda	5.923355	255	145
intron	-5.8909	36	104
transcript	-5.87735	259	391
substrat specif	5.871841	75	19
hybrid	-5.84328	112	211
rt	-5.81242	20	76

Appendix F

Implementation Notes

This Appendix describes the various file types and perl scripts we used to process our datasets, train and test our classifier and calculate the *Z-score* and Kullback-Liebler divergence between term distributions. The perl scripts discussed here, as well our datasets and examples of the different file types, are available on a server called `redtape`, in the `/fs/hs/projects/CharizedProts/FinalCode/` directory. The `redtape` server is part of the Queen's University School of Computing's network.

F.1 Text Processing

We downloaded an XML copy of the PubMed database in March 2008. Note that the entries for some PMIDs may appear twice in the PubMed download. The PubMed download is stored on `redtape` in the `/fs/rt1/PubMed/` folder.

We used the following scripts to scan the PubMed download to collect articles of interest in XML format, convert the XML to a simpler format called `.itame`, and then convert the articles in `.itame` format into `.stat` format, which is the binary term representation that we use with the classifier (as discussed in Section 4.1). The next paragraphs describe the perl scripts and file formats we employed. The scripts described in this section can be found in the subfolder: /`FinalCode/TextProcess/` .

`getXMLforPMIDs.pl` – **Input:** Name of PMID file, name of output file. **Output:** Writes XML to output file, writes PMIDs that were not found to `notFound-<outFile>.txt`, writes status to the command line.

This script takes a list of PubMed IDs (PMIDs) and then scans the PubMed download in order to find the XML entries and write them to an output file. Each PMID should be on its own line in the PMID file. The location of the PubMed download is hardcoded on line 3 of the script.

`XML2ITAMERob.pl` – **Input:** optional `-a` flag to require abstracts, name of XML file, name of output file. **Output:** Writes `.itame` to output file, writes status to command line.

This script takes an XML file and converts it to `.itame` format (discussed next). There is an optional `–a` flag in the input to specify whether to include or omit articles that lack abstracts (which is possible in PubMed entries). If the `-a` flag is present, only articles with abstracts are converted. Note: it is important to run the `removeDupesFromITAME.pl` script (described below) on each `.itame` file generated from PubMed in order to remove any duplicate `.itame` entries.

`.itame` format – In this format, the PMID, date, title, and abstract of each article exists on its own line, and is preceded by .I, .D, .T and .A respectively. Each entry is delimited by .E. Two `.itame` file examples are included in the `/FinalCode/Examples/` folder: `jolPos.itame` and `jolNeg.itame` .

`removeDupesFromITAME.pl` – **Input:** Name of `.itame` file **Output:** Writes unique `.itame` entries to the command line.

This script outputs each `.itame` entry from the input file to the command line (so it can be piped into a new file). It only outputs one `.itame` entry per PMID – duplicate entries will not be outputted. It is necessary to process newly generated `.itame` files with this script in order to ensure that they do not contain duplicates.

`stemITAMEwithTermSet.pl` – **Input:** Name of `.itame` file, optional `-t, -p,` and `-l#` term set flags. **Output:** Writes `.stat` file to the command line.

This script takes the name of an `.itame` file as input, along with any combination of three flags. It stems terms by default and removes stop words, email addresses, web addresses and numbers. Stemming is performed using Porter's Stemmer [49]. We evaluated using several different sets of terms to represent articles (as described in Sections 4.1 and 5.1). Unigrams are included by default. Including the `–p` flag as input causes the script to include bigrams as additional terms. Bigram term are appended with a `!P`. The `–t` flag causes terms in the title to be counted twice: once as normal terms and once as title terms. Title terms are appended with `!T`. Finally, the `–l#` flag causes the last # terms in the abstract to be counted twice: once as normal terms and once as *late abstract* terms. The # should be replaced with an integer (`-l5` or `–l10`, for example). This controls how many terms at the end of the abstract are counted twice. Late abstract terms are appended with `!L`. The term

representation for each document is outputted as a `.stat` file (discussed next) to the command line, where it can be piped to a file.

`.stat` format – The `.stat` file format is used to hold the term representation of the articles in a dataset. It is a very simple format: the entry for each article begins with its PMID and a colon, which is followed by each of the terms that occur in the articles. The PMID and each term are on their own lines, and article entries are separated by blank lines. Two `.stat` file examples are included in the `/FinalCode/Examples/` folder: `jolPos.stat` and `jolNeg.stat` .

F.2 Datasets

Each of our four initial datasets (as described in Chapter 3) and the ***SPonly*** dataset (described in Sections 5.4.2 and 5.5) are available in the `/FinalCode/Datasets/` folder in both the `.itame` format, and the `.stat` format. The `.stat` files were created using the default feature selection approach as described in Table 5.1. Each dataset is broken into two files: one containing the *relevant* articles, which is labeled with '`Pos`', and another containing the *irrelevant* articles, which is labeled with '`Neg`'. Table F.1 shows the filenames for each dataset.

Table F.1: Dataset Filenames

Dataset Name	Section Discussed	Filenames
Curated Journal	Section 3.1	`curJolPos` and `curJolNeg`
Curated Swiss-Prot	Section 3.2	`curSPPos` and `curSPNeg`
SP-GO	Section 3.3	`spGOPos` and `spGONeg`
CHAR	Section 3.4	`CHARPos`
SPonly	Section 5.4.2 and 5.5	`spOnlyPos` and `spOnlyNeg`

F.3 Classification

The following paragraphs describe the file formats and perl scripts we used to train and test our naïve Bayes classifier. The final script discussed in this section, `testSuite.pl`, is capable of applying our feature selection approaches (as described in Section 4.2) as well as performing a cross-validation of the classifier. There are alternate versions of the `testNaiveBayes.pl` and `testSuite.pl` scripts that employ the only-present-terms calculation (which was introduced in Section 4.4.2). The following files can all be found in `/FinalCode/Classfy/`, unless otherwise noted.

`trainNaiveBayes.pl` – **Input:** Name of `.stat` file containing *positive* articles, name of `.stat` file containing *negative* articles. **Output:** Writes naïve Bayes classifier to the command line.

This script trains a naïve Bayes classifier from a `.stat` file of *positive* training articles and a `.stat` file of *negative* training articles, as described in Section 4.4.1. It calculates the prior probability for each class and the conditional probability for

each term given the class and outputs the results to the command line so that it can be piped into an `.nb` file (described next).

`.nb` format – The `.nb` format stores all the probabilities needed for naïve Bayes classification. It is a simple format, with the line `.Positive Conditional Probabilities` preceding the positive probabilities and `.Negative Conditional Probabilities` preceding the negatives. The prior probabilities are prefaced with `.Prior:`, and the conditional probabilities for each term are prefaced with the `term:`. An example `.nb` file, named `jolClassifier.nb`, is included in the `/FinalCode/Examples/` folder.

`testNaiveBayes.pl` – **Input:** Name of `.nb` classifier file, name of `.stat` file. **Output:** Writes PMIDs from `.stat` file and their predicted class to the command line.

This script takes the probabilities in the supplied `.nb` classifier file and uses them to predict the class for the articles in the supplied `.stat` file. PMIDs from the `.stat` file are outputted to the command line along with the class predictions, as well as the calculated probability for the article to belong to each class. The program also prints a summary that shows the total number of articles that were assigned the *positive* or *negative* class.

The `testNaiveBayesOnlyPres.pl` script behaves identically to this script, with the exception that it uses the only-present-terms calculation (Section 4.4.2) to determine the class for each article.

`testSuite.pl` – **Input:** Number of cross-validation folds (*k*), name of `.stat` file containing *positive* documents, name of `.stat` file containing *negative* documents, optional `–minFreq#` and `–maxFreq#` flags for document frequency filtering, optional `–z#` flag for *Z-score* filtering, optional `-specRemove` or `–specReplace` flags to remove or replace species-specific terms (requires `ncbiSpeciesApr14.txt`). **Output:** Writes a `.nb` classifier to the command line if $k = 1$, writes cross-validation results to the command line if $k > 1$, writes status to command line using perl warnings.

This script is capable of performing three different types of feature selection on the terms in the supplied `.stat` files before training a naïve Bayes classifier (as described in Section 4.2). It is also capable of performing a standard *k*-fold cross-validation. The first input value (*k*) controls the behavior of this script. If *k* is set to 1, the script will perform feature selection as specified and then output the classifier it trained to the command line in `.nb` format. However, if *k* is set to an integer greater than 1 then the script will perform a *k*-fold cross-validation of a classifier trained using the `.stat` files and employing the feature selection methods as specified. In this mode, the results of the cross-validation are outputted instead of the `.nb` classifier.

The `–minFreq#` and `–maxFreq#` flags control document frequency feature selection (Section 4.2.1). The # can be replaced with either an integer >= 1, or a fraction < 1 (e.g. 0.6). If the # is an integer, then the script will remove terms that

occur in fewer documents than the integer (for minFreq) or terms that occur in more documents than the integer (for maxFreq). The default minFreq is 3; terms that occur in fewer than 3 documents are removed if no `–minFreq#` is specified. If the # is a fraction < 1, then the script will remove terms that occur in fewer than that fraction of the documents (for minFreq) or terms that occur in more than that fraction of the documents (for maxFreq). The default maxFreq is 0.6; terms that occur in more than 60% of the documents are removed if no `–maxFreq#` is specified.

The `–z#` flag controls *Z-score* filtering (Section 4.2.2). The # is the cutoff *Z-score* value (e.g. `–z3.2`). Terms with a *Z-score* below the # are excluded from the classifier. The default behavior (if no `–z#` flag is specified) is to not remove any terms based on *Z-score*.

The `-specRemove` or `–specReplace` flags control the handling of species-specific terms (Section 4.2.3). The `-specRemove` flag will cause species-specific terms to be removed from the feature set, while the `–specReplace` flag will cause these terms to be replaced with a common placeholder. This feature selection method only acts on terms that are specified in the file `ncbiSpeciesApr14.txt`, which is included in the `/FinalCode/Classify/` folder, and is described next.

The `testSuiteOnlyPres.pl` script behaves identically to this script, with the exception that it uses the only-present-terms calculation (Section 4.4.2) to determine the class for each article.

`ncbiSpeciesApr14.txt` – This file is a list of species-specific terms that occur in the datasets. As discussed in Section 4.2.3, we downloaded a list of species names from NCBI and then manually confirmed the terms that occurred in the datasets in order to avoid acting on a term that is not species-specific. Note that this is *not* a comprehensive list of species terms.

F.4 Statistics

The following two scripts were used to calculate the *Z-scores* and Kullback-Leibler divergences we discuss in this thesis. The can be found in the `/FinalCode/Stats/` folder.

`zTest.pl` – **Input:** Name of `.stat` file containing *positive* documents, name of `.stat` file containing *negative* documents. **Output:** Writes the *Z-score* for each term to the command line.

This script calculates *Z-score* as we described in Section 4.2.2. In addition to outputting the *Z-score* for each term, this script also prints the number of times the term occurs in the *positive* documents and the number of times the term occurs in the *negative* documents. The format of the output is `<# occurrences in pos`

docs>-<# occurrences in neg docs>:<term>:<*Z-score*>. The output is sorted by *Z-score*.

KLdivergence.pl – **Input:** Name of `.stat` file containing one dataset, name of `.stat` file containing another dataset, name of file containing list of terms. **Output:** Writes the Kullback-Liebler divergence between the term distributions of the two datasets in the `.stat` files to the command line.

This script calculates the Kullback-Leibler divergence between two datasets, over the term set defined in the term list file, as discussed in Section 5.4. The output is a single number that represents the divergence between the datasets. An example of a term list file is included in the `/FinalCode/Examples/` folder. It is named `jolPosNeg.termList`.