Abstract

A method for automatic identification of PubMed abstracts discussing functional aspects of a given gene is presented. This identification task is addressed as a combination of document retrieval followed by subsequent classification. The *Retrieval* problem involves finding documents that are relevant to particular genes; we formulate and use a set of rules for query expansion to improve the recall. The *Classification* problem is addressed from a supervised machine learning perspective and solved using document classification techniques, using both the abstract text as well as domain knowledge derived from existing ontologies. Several different variations of the basic approach have been tried and evaluated in terms of precision and recall on document collections of different sizes. We conclude with a discussion of the results and their interpretation with respect to the information used in each case.

Contents

List of Figures

Acknowledgements

I would like to thank my advisor Prof. Marti Hearst for all the discussion, ideas and help in moulding this work and this report. Her guidance was invaluable in the realization of this work.

I would also like to thank Preslav Nakov for his text processing code, related work that was taken out of the term project that we did together and useful ideas that germinated during the course of our discussions. In addition, I would like to thank Ariel Schwartz for his help in downloading and maintaining the databases and installing the DB2 text search module. Finally I would like to thank IBM for allowing the research use of DB2 and their Text Search module. This material is based upon work supported in part by the National Science Foundation under Grant No. EIA-0122599.

Chapter 1

Introduction

Recently, a large number of genetic discoveries have been reported in the bioscience (molecular biology, biomedicine, medicine, etc.) literature. An enormous amount of data is available via websites maintained by the National Center for Biotechnology Information (NCBI) [3] and by other research groups around the world. This wealth of electronically published knowledge can be used by scientists and clinicians seeking information in a new area or wanting to find information on a particular topic. One such database is PubMed ([6], referred also as Medline), which nowadays contains $6.6M$ plus abstracts over a vast corpus of bioscience literature, and is growing by more than $40K$ citations each month. While some part of the document representation in PubMed is structured, much is in the form of article abstracts.

Finding information relevant to one's interest without being overwhelmed by all the non-relevant information is a daunting task. This can be addressed using automated methods to integrate raw and processed data from variety of sources, and evaluate and digest the contents to present them in a more usable format. With the multi-fold increase of the size and complexity of biological information [30], such automated methods also become critical to assist human operators in collecting, curating and maintaining the various databases. An example of a database that could benefit from automated curation methods is LocusLink [5]. LocusLink contains PubMed references for documents related to genes and their functions. We describe LocusLink in detail in later chapters.

Swanson and Smalheiser [44] demonstrated the potential of literature text mining, by discovering an important but previously unknown relationship between *magnesium* and *migraine headaches* by using simple word cooccurrence statistics in article titles. This success inspired researchers from the machine learning and information retrieval community to do further work in the application of text mining and natural language processing techniques to biomedical texts (e.g., Hearst [24]). The bioinformatics community is increasingly applying machine learning, natural language and information retrieval tools to the bioscience literature. Current tasks of interest include named entity recognition, terminology disambiguation, gene protein interaction identification, relations extraction, etc. We describe the state of the art in some of these in our related work section.

In this report we present a method that attempts to find, for a gene name or synonym, a set of articles or documents that talk about some of its functional aspects. This work is inspired by the GenomicsTrack [1] of the Text REtrieval Conference (TREC) [8]. TREC is an IR forum that is sponsored by the National Institute of Standards and Technology (NIST) and several other government agencies. It was designed to encourage and support large scale information retrieval research. Each year TREC consists of a main retrieval task and a number of specialized tracks. The GenomicsTrack consists of two tasks, the first one requires ad hoc retrieval and the second one requires information extraction from the retrieved documents. Our work focuses on the first task that consists of finding all journal articles in a collection of documents that describe some aspect of a function related to a particular gene belonging to a given organism.

This work consists of two parts. In the first part we present a Classification Module, where we identify the documents that talk about a *function* of a given gene. We address the problem from a supervised machine learning perspective and solve it using document classification techniques. We identify various features in the input documents using both the abstract text as well as domain knowledge derived from existing ontologies. These features are then converted into a feature vector and fed into a Naive Bayes [33] classifier to learn a model which is used subsequently to predict the class of a new document.

In the second part we present a Retrieval Module, where we identify the subset of documents that are associated with a given gene. An easy way to do this would be to use the information contained in online databases like LocusLink[1], but these databases are not exhaustive and contain only a small subset of documents related to the gene. We obtain a more complete set by searching for the gene name and synonyms in the PubMed documents. The

[1]LocusLink contains references to relevant journal articles called PubMed references for some of the genes

task of retrieving documents related to a gene is non-trivial since the gene names may be used in the documents in various different lexical forms and the same names may be used for genes that belong to different organisms.

Finally we show how these two systems could be combined together to perform the trec task that was described before. We use the classifier to assist the retrieval module in ranking the documents. The scores from the classifier, that denote the likelihood of documents belonging to one of the classes (being function references or not), are interpreted as relative relevance of the documents and used in ranking them. We find that this combination achieves only a marginal improvement in performance.

The rest of the report is organized as follows: In chapter 2 we describe related work. We talk about our Classification module in chapter 3, where we describe how we filter documents that are potential Gene References into Function. In chapter 4 we talk about our retrieval module and how we apply the classifier to the retrieved documents to obtain final results for our task. Finally, in chapter 5, we conclude the discussion with the summary and implications of the results and outline possible future work.

Chapter 2

Related Work

2.1 Gene name identification

An important step in every text processing algorithm is the identification of the terms of interest: the so-called *named entities*, according to the terminology adopted in information extraction. For the case of *protein* identification the problem has been proved easy. Probably the most sophisticated identification of *protein* names is due to Fukuda et al. [20], who propose a rule-based approach and achieve 94.70% precision and 98.84% recall. The problem has also been addressed by Proux et al. [37], who are interested in *gene* name identification and apply a combination of rules and statistics (precision 91%, recall 94%). Collier et al. [14] extract both *genes* and *gene products* names using hidden Markov model (F-measure 73%). Tanabe and Wilbur [47] apply tagging. Part of Speech (POS) tagging as well as phrasal chunking is used by the EDGAR system [39].

Next step after identifying the named entities is to map them to a known *gene* or *gene product.* This mapping can be difficult because there is no accepted standard way of representing these names. Naming conventions do exist for some types of biological concepts, e.g., genes, alleles and proteins [35]. There are formal bodies, such as Enzyme Commission and Human Gene Nomenclature Committee, whose responsibilities involve assigning unique symbols and descriptive names to specific types of concepts. Still, these are only guidelines and as such do not impose restrictions to domain experts. In addition they only apply to a subset of terms, while the rest of the bioscience terminology remains highly non-standardized.

At present, we do not address the identification problem in full, but rather just perform a mapping against a predefined list of genes and other biological terms from the MeSH hierarchy. We also identify syntactic variations of gene names by matching known names against all possible subsequences of words that appear in actual text (see chapter 4). Below we discuss related work in two tasks that help with the above mapping, *synonym extraction* and *terminology disambiguation.*

2.1.1 Synonym Extraction

Most approaches to synonym identification work with the content of the strings representing *gene* and *protein* names. Humphreys and Lindberg [27] used a semi-automatic method to identify multi-word synonyms in UMLS (the Unified Medical Language System, a large biomedical taxonomy) by linking terms as candidate synonyms if they shared any words (Hole and Srinivasan [25]). E.g. the term *cerebrospinal fluid* leads to *cerebrospinal fluid protein assay.* The candidate synonym terms were then evaluated by human curators. This approach is tedious for a large scale system due to the need for human marking of results. Hanisch et al. [22] used a token model of protein names to construct a synonym list. Their assumption is that the order of words in names does not matter and they partition words into token classes according to semantic significance. Given a protein name they find the list of candidate synonym sets based on the token classes that match and then assign the names to the set with the best score. A scoring scheme is learned using Robust Linear Programming. They achieve a specificity of 0.78 and sensitivity of 0.82 without curation on a set of 611 associations (141 objects in 470 abstracts).

Yu and Agichtein [51] use the surrounding context of gene and protein names to identify synonyms. They explore four complementary approaches for discovering contextual similarity for words in text, namely unsupervised, partially supervised, and supervised machine learning techniques, as well as a manual knowledge-based approach. They retrieve only synonym pairs that appear together in a sentence. Their best system combines these approaches and gives a very low precision of 0.15 at 0.5 recall on a set of 588 synonym pairs. Wilbur and Kim [48] employ a trigram-matching algorithm to identify similar multi word phrases. They treat phrases as documents made up of character trigrams which are then represented in the vector space model and similarity computed as the cosine of the angle between the corresponding vectors. We use a similar approach to identify lexical variations of gene names, which are then used to generate rules for matching.

2.1.2 Biological terminology disambiguation

Another important problem related to gene name identification is ambiguity. It is common in the biological literature to use the same graphemic form to refer to *gene*, *protein*, *gene transcripts* (mRNA, rRNA, tRNA) etc. The problem can be addressed as a word sense disambiguation (document classification) task. Liu et al. [32] propose an automatic method that constructs sense-tagged corpora for ambiguous abbreviations in the UMLS using Medline abstracts. They achieve overall precision of 92.9% and an overall recall of 47.4% on a set of 35 frequently occurring ambiguous biomedical abbreviations. Hatzivassiloglou [23] tried Naive Bayes, decision trees and RIPPER, exploiting various features (position, capitalization, POS, word distribution, stop-words removal and stemming) and achieved accuracy of $78-84\%$ over a set of 550 manually labelled ambiguous occurrences.

2.2 Gene interactions identification

Gene function can be can expressed as a binary relation in several different ways, including: gene-gene, gene-protein, gene-drug, gene-disease, gene-process etc. We will describe some interesting approaches to extracting them below, as well as some related ones: e.g., protein-protein interactions, which do not describe a function of a gene.

2.2.1 Gene-protein interaction

Sekimizu et al. [42] have adopted an elaborate and linguistically motivated approach to this task. They use a list of predefined verbs (*activate, bind, interact, regulate, encode, signal, function* etc.) and perform a shallow parsing using EngCG to identify the interacting proteins. They infer that genes and/or gene products interact only in those cases in which they are the subject and the object of that verb. (An experiment for the verb *activate* shows that about 65% of its arguments are genes or proteins.) A precision of 90% is reported for the noun phrases bracketing (using regular expressions over the output of EngCG) and 73%, for the subject and object recognition given the target verb. The method is further extended to handle conjugate forms: e.g., *activate - activation, activate, activates, activator, inactive, ...-activated, ...-activating.* Although the system is described as the first step toward building genome-related thesaurus and hierarchies in a fully automatic way, it is fully functional and has been formally evaluated: the precision of interaction identification is between 67.8% and 83.3% for the different verbs. No recall information is provided.

2.2.2 Gene-drug interactions

Kamvar et al. are interested in extracting gene-drug relationships from biomedical literature [29]. They address the problem from a supervised machine learning perspective. Their algorithm uses a training set of known gene-drug interactions and representative literature about each gene and drug as a training set and tries to identify new unknown interactions that cannot be inferred trivially (e.g., by lexical similarity between a new gene/drug and one in the training set). The relation is predicted for a gene-drug pair G2-D2 (from the literature) for which there is a pair G1-D1 in the training set such that G1 and G2, as well as D1 and D2, are similar enough. The *probability* of interaction P(int(G2,D2)) between G2 and D2 is estimated as $P(int(G2, D2)) = P(int(G1, D1)) \times sim(G1, G2) \times sim(D1, D2)$. A vector space model is used to estimate the similarity between two genes or drugs. Each gene and drug is assigned a vector based on the words that appear in their description in literature (they use LocusLink[5] for genes and USP DI[9] for drugs descriptions). The similarity is then calculated as a dot product between the corresponding normalized vectors (the cosine). The approach is carefully evaluated on a set of 258 gene-drug pairs and a precision of 60% is reported (which is amazing, given its simplicity), but no recall is calculated.

2.2.3 Gene-gene functional similarity

Shatkay et al. [43] try to identify functionally similar genes from biomedical text using information retrieval techniques. Given a list of genes and a starting (kernel) document for each, they find 50 additional documents that are most similar along with a list of characteristic words. The idea is that if the kernel document is used to discuss the functional role of a gene, so will the highly similar ones. Thus two genes are considered similar if the corresponding sets of 50 documents are similar, which is measured in a standard way: as the cosine between the corresponding vectors. They report some preliminary experiments but do not perform any objective evaluation of their technique.

2.2.4 Protein-protein interaction

Blaschke et al. [11] describe a system for automatic extraction of protein-protein interactions from PubMed [6] abstracts by means of pre-specified lists of protein names and 14 verbs (*acetylate, activate, associated with,* *bind, destabilize, inhibit, interact, is conjugated to, modulate, phosphorylate, regulate, stabilize, suppress* and *target*), indicating actions related to genes interactions. They look for instances of a pattern of the form protein-verb-protein, all three met in the same (part of the) sentence and in that order (with possibly some other words in between). The extraction is performed over a set of documents and only the most frequent instances are retained, the rest being ruled out as likely to be due to chance. The method has not been quantitatively evaluated but has been applied over a set of proteins, where it recovered most of the known interactions (and suggested some additional unknown ones). Although it has been proposed for protein-protein interactions, it would be easily adaptable to gene-protein and gene-gene interactions.

Thomas et al. [45] address protein-protein interaction from an Information Extraction (IE) perspective: they adapt the general-purpose IE engine HIGHLIGHT[1], which allows them to build elaborate contextual rules, combining syntactic and lexical information. They developed mapping rules for over 30 verbs, but concentrated on just three of them: *interact*, *associate* and *bind*. They use *frequency* of occurrence of relations, the *context* in which a relation is found and the *confidence* that the NP arguments are really proteins to score the rules. The reported precision varies from 60% to 81%, but the recall is low: from 24% to 63%.

2.2.5 Other Interactions

An interesting IE system is PASTA[21]. It extracts information concerning the roles of amino acids in protein molecules. It identifies and tags 12 different kinds of biological named entities (protein, species, residue, site, region, secondary structure, supersecondary structure, quarternary structure, base, atom, non-protein compound and interaction), then classifies these into three template elements (RESIDUE, PROTEIN and SPECIES) and looks for two templates (IN_PROTEIN and IN_SPECIES). The entity recognition task proceeds in three steps: *morphological analysis* (identifies if individual token have interesting biochemical affixes), *lexical lookup* (for term type classification) and *terminology parsing* (combination of component terms into single multi-token unit using a rule based parser). POS tagging is then used to convert sentences into predicate-argument expressions, which are subsequently combined to generate templates. The precision/recall figures for named entity, template element and template relation are: 84%/82%,

[1]http://www.cam.sri.com

66%/75% and 65%/68%.

Other interesting natural language processing (NLP) techniques applied to the field of gene and gene product relations extraction use preposition-based shallow parsing (Leroy and Chen [31], precision 70%, recall 47%), categorical grammar parsing (Park et al. [36], precision 80%, recall 48%) and XHPSG parsing (Yakushiji ([49]). Pustejovsky et al. [38] go even further: in addition to performing shallow parsing and using domain knowledge (UMLS ontology), they use an anaphora resolution (sortal and pronominal) module to boost the relation extraction accuracy.

2.3 Classification

One of the tasks we are interested in is to identify if a given document discusses gene function. Since we are *not* interested in the *specific* relation/function but just in the fact of its existence we can address the problem as a document classification task.

Classification is a basic machine learning task. It uses a two-stage process: training and testing. During training the program is provided with a set of labelled examples which are used for learning algorithm parameters. While testing the system is asked to predict the class for unseen example(s). A broad range of algorithms have been used for classification, including: Naive Bayes [19], decision trees, decision lists, support vector machines, neural networks, k-nearest neighbor, Rocchio etc. (see [33]) Classification algorithms are useful for NLP tasks that can be formulated as classification/discrimination problems, e.g., word sense disambiguation and in particular, biological terminology disambiguation that we discussed in section 2.1.2.

Other tasks, like relations extraction can also be reformulated as a classification problem. Rosario and Hearst [40] identify semantic relations between constituents in bioscience text. They identify a set of eight possible relations that can hold in natural language sentences between entities *treatment* and *disease*. They use neural network to classify these entities/relations and achieve 96.9%/76.5% accuracy in relationship extraction when entities are given/hidden. Craven and Kumlien [15] also address the relation extraction from a text classification perspective. They work at the sentence level and learn a Naive Bayes classifier which predicts whether a specific sentence is an instance of the target relation, e.g., *associated-diseases(Protein, Disease).* We follow a similar approach, but at the *document level*: given a set of labelled example documents we learn classifiers to predict whether a new document, containing a given gene name, discusses about some function of that gene. We are not interested in the particular function type but rather in the single fact that it is being discussed there.

Chapter 3

Classification Module

Our approach to document classification is based on the idea that articles which discuss gene function contain a distinct set of features which can be learned using automated techniques. The resulting models can be used to classify new documents.

The LocusLink[5] database from NLM contains information about gene functions. This information can be collected via a handcrafted crawler, but more recently it has been made available as a down-loadable database. Each LocusLink record refers to a set of PubMed documents that are associated with the gene that talk about its mapping, identification or function (PubMed references). A subset of these are annotated as GeneRIFs (Gene Reference Into Function), i.e., they talk about a functional aspect of the gene in question. Each GeneRIF consists of a short phrase describing one or more functions of the gene, and a PubMed identifier (PMID) that identifies a citation in the PubMed database for a published paper that describes this function. For example, "amyloid beta (A4) precursor protein" (LocusID-351) that is a biomarker for Alzheimer's disease contains 83 PubMed references and 60 GeneRIFs; two of the GeneRIFs are:

11500807	*Mutations in APP may predispose to very-late-onset Alzheimer disease*
11915326	*forms a seed of amyloid beta protein aggregation via binding to G(M1)ganglioside*

We describe in the following section how we use this information to generate training and testing sets for our classifier.

3.1 Preparation and Preprocessing

We first obtain the genes for which there is a GeneRIF entry in the LocusLink database (Some of the genes do not have either of the references described above). For each such gene we collect the documents annotated in the GeneRIF field of the LocusLink record. We have a local copy of the PubMed database created for the purpose, which contains bibliographic and other information for over $12M$[1] documents from $4,600$ journals. These GeneRIFs serve as the set of positive instances for our algorithm. The remaining PubMed documents for which there is a reference from LocusLink, but no annotation with GeneRIF, serve as the negative set for our algorithm. The assumption is that since the documents are referenced from LocusLink, they discuss something about the gene; but since there was no GeneRIF annotation, they do not discuss function. The GeneRIF references are not exhaustive since their annotation in LocusLink is done by manual curation, there is a possibility that some of the function references have been missed, i.e., there could be false negatives in the database. We found some of the documents belong to both the positive and the negative sets, and we removed them (they constitute less than 1% of total) because they may reflect possible human errors in annotation.

Next we extract the set of features we will use. Once the features have been identified, they are fed into a classifier in order to learn to classify new documents, i.e., to recognize whether or not they talk about the functional aspects of the target gene.

We do not have full text for most of the PubMed articles, so we have somewhat restricted our scope to consider a scientific genre involving abstracts, together with other important features that have been extracted out of an annotated set of documents. Fortunately, PubMed contains a good collection of human curated information that can be used as features. Once we have identified the features that are to be used for the given documents; we need to preprocess them. This step generates a feature vector representation of the documents that appear as instances of the two classes. The specific features used are described below. We also experiment with combining different feature sets.

[1]Out of this $6.6M$ citations also have the abstracts from the corresponding journal article.

3.1.1 Abstracts from the documents

Abstracts generally summarize the content of the document or article they belong to, and contain useful information that are features for gene function information detection. To speed up the processing we remove all the stop words from the abstracts. Stop words are commonly occurring words in text that do not discriminate the content. E.g., words like *a*, *the*, *as*, *to*, etc. are common across both the positive and the negative classes. One method to identify stopwords would be to generate domain specific words that appear in more than 75% of the documents. This method of identifying stop words has been used with success in similar applications [12, 50], but in our case it did not work well. We did not find many words, beyond those that are known standard stop words, common to more than 50% of the set ($2,000$ abstracts). We ran a small experiment to count the frequency of the most common words. Those with frequency $> 50\%$ are shown in figure 3.1. The only word corresponding to the bioscience domain is “protein” which we add to our stoplist. The lack of domain specific stopwords may be explained by the fact that we use *abstracts only* (as opposed to full documents), which are short enough so that the chance for any two of them to share a particular word (even a stop-word) is reduced dramatically. Moreover abstracts may differ from full text in sense that they might use an abbreviated and more compact style of representation. So we use a standard list of around 321 stop words commonly used in Information Retrieval with the word “protein” added to it. After removing the stop words we also remove those words that appear less than a given frequency across the whole set of documents. This number is a parameter for the preprocessing module and is currently set to 5. The intuition behind removing the less frequent words is that this prevents the model from over-fitting to the training set.

Next, for each document in the training and testing set we construct word vectors that have as dimensions the distinct words found across the combined set. Each dimension for individual document vector is assigned the value 1 if the corresponding word is present in the abstract for the document, else it is assigned the value 0.

3.1.2 MeSH Headings

We use NLM’s[4] controlled vocabulary, Medical Subject Headings (MeSH [2]), to characterize the content of the articles represented by PubMed citations. These headings appear in the form of descriptors and qualifiers for those descriptors (they are present in PubMed as annotation for each cita-

Word	No. of documents	Frequency in %
the	1875	93.75
of	1873	93.65
and	1861	93.05
in	1851	92.55
to	1747	87.35
that	1633	81.65
is	1508	75.40
with	1429	71.45
we	1414	70.70
by	1377	68.85
for	1320	66.00
protein	1018	50.90
this	1008	50.40

Figure 3.1: Number of Different abstracts each word appears in

tion it contains). For example the PubMed document titled "The importance of an innervated and intact antrum and pylorus in preventing postoperative duodenogastric reflux and gastritis" (PubMedID 123), contains 19 descriptors (*Adult*, *Bilirubin*, *Duodenal Diseases*, *Duodenal Ulcer*, *Endoscopy*, *Female*, *Gastric Juice*, *Gastric Mucosa*, *Gastritis*, *Heartburn*, *Human*, *Hydrogen-Ion Concentration*, *Male*, *Middle Age*, *Postoperative Complications*, *Pyloric Antrum*, *Pylorus*, *Stomach Diseases*, *Vagotomy*).

Qualifier names are known as subheadings, which, when combined with the descriptor names, form the central concept of the article. Some descriptors do not have associated qualifier names when the heading alone is the central concept of the article. E.g., amongst the descriptors in the above example, *Adult*, *Endoscopy*, *Female*, *Human,Hydrogen-Ion Concentration*, *Male*, *Middle Age* and *Vagotomy* have no associated qualifiers, *Duodenal Diseases* and *Gastritis* have two qualifiers each ([*prevention & control*; *radiography*] and [*pathology*; *prevention and control*] respectively) while the rest of them have a single qualifier each. We use the descriptors alone and in combination with the qualifiers as different features for document representation. As before the feature vectors for documents contain these MeSH descriptors and qualifiers as individual dimension, each dimension being 1 if the corresponding descriptor/qualifier is present and 0 otherwise.

3.1.3 MeSH Hierarchy Tree Numbers

MeSH consists of concepts in the form of descriptor terms or headings arranged in a hierarchical structure that permits searching at various levels of specificity. At the most general level of the hierarchical structure are very broad headings such as "Anatomy" or "Mental Disorders." At more narrow levels are found more specific headings such as "Ankle" and "Conduct Disorder." We downloaded this information from NLM into a locally maintained database. We added the associated tree labels to the feature set for each of the descriptors in PubMed for a given document. We also experimented with cutting the tree labels at various hierarchy levels in order to understand how the specificity of MeSH terms affects the learning. In MeSH, each concept is assigned one or more alphanumeric descriptor codes corresponding to particular positions in the hierarchy[2]: e.g. *A* (Anatomy), *A01* (Body Regions), *A01.456* (Head), *A01.456.505* (Face), *A01.456.505.420* (Eye). Eye is ambiguous according to MeSH and has a second code: *A09.371* (*A09* represents Sense Organs) [34]. The mapping ambiguity could potentially introduce additional noise when the tree labels are used as features.

3.2 Training

Once the feature vectors are obtained, we train a classifier to build a model that can predict whether a document talks about gene function. We use a Naive Bayes classifier [19, 33] to train the model. Its fundamental idea is the assumption that the values of the feature variables $F = (F_1, F_2, ..., F_n)$ are conditionally independent given the class variable S. The joint probability is given by the expression:

$$p(S, F) = p(S) \prod_{i=1}^{N} p(F_i|S) \tag{3.1}$$

The model parameters are given by the probabilities $p(S)$ and $p(F_i|S)$, which are usually estimated from the text by means of maximum likelihood estimates (MLE). The classification of a new concept is determined by the most likely category:

$$S_{ML} = arg \max_{S_k} p(S_k|F) \tag{3.2}$$

[2]There are 15 main categories or branches in the MeSH hierarchy, these branches then have others branching off them and so on with each branch becoming more and more specific. The current hierarchy is nine levels deep and there are a total of 39824 nodes over all the levels.

Naive Bayes classifiers are among the most successful known algorithms for document classification. The Naive Bayes classifier is known to be optimal when attributes are independent given the class, but Domingos et al. [18] show that it will often outperform more powerful classifiers for common training set sizes and numbers of attributes even if the independence assumption is not met.

The implementation of the Naive Bayes classifier we currently use is part of the open source machine learning package called WEKA (Waikato Environment for Knowledge Analysis [10, 28]) from the University of Waikato, New Zealand. They provide excellent Java implementation of Naive Bayes and other machine learning algorithms.

3.3 Evaluation Method

We evaluate the model generated during training using a ten fold stratified cross-validation. We split the data into 10 sets using a random number generator with a fixed seed. These partitions are then combined to generate 10 different train/test sets: 90% for training and 10% for testing.

We obtain the average values for the number of correctly classified instances from cross-validation. We also obtain a confusion matrix giving the number of *True Positives (TP)*, *True Negatives (TN)*, *False Positives (FP)* and *False Negatives (FN)* averaged over the 10 runs. *Precision* and *recall* are easily computed using these values. $Precision(P)$ is the ratio of the number of relevant documents over the total number of documents retrieved. It is a measure of the amount of useful information produced by the system. $Recall(R)$ is the ratio of relevant documents retrieved over the number of relevant documents in the database. Recall is a measure of how completely the set of relevant documents were retrieved. P and R are computed as follows:

$$\begin{aligned} P &= \frac{TP}{TP+FP} \\ R &= \frac{TP}{TP+FN} \end{aligned} \tag{3.3}$$

Having more complete results comes at the cost of more non-relevant documents in the result set, and having less non-relevant documents in the result set tends to be associated with less complete retrieval of the intended documents or results. So, there tends to be a trade-off between precision and recall. Different measures have been proposed in the literature to combine them into a single measure based on their relative importance. Van

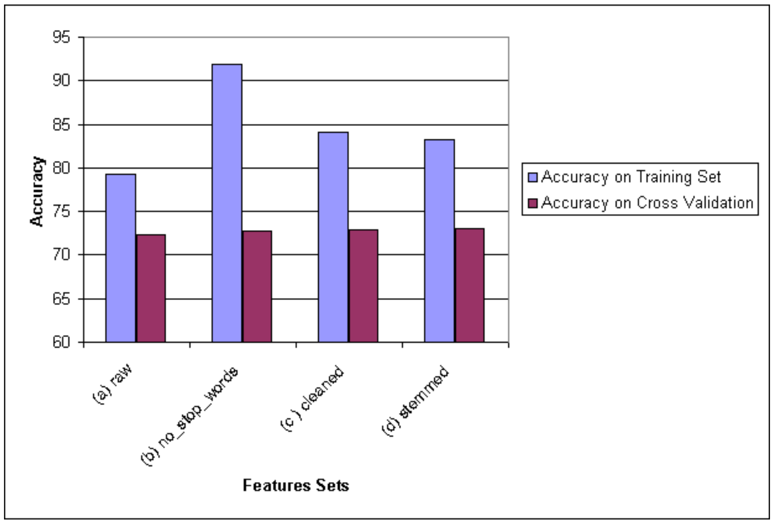

Figure 3.2: Classification accuracy with different methods of preprocessing abstracts: (a)Without Preprocessing (b)Removing Stopwords (c)Removing less frequent words in addition (d)With addition of stemming

Rijsbergen [46] defines and justifies the so-called *F-measure* as follows:

$$F_b = \frac{(b^2+1)PR}{b^2P+R} \tag{3.4}$$

We use $b = 1$, i.e., recall and precision are equally weighted.

3.4 Experimental Results

We use 1000 instances from each of the positive and negative classes as input to the algorithm. We simply choose the first 1000 documents for each class from the sets created before, see section 3.1 (we later show how the accuracy varies with the training set size). In the first experiment the abstracts are fed straight into the system, without any preprocessing. Next we remove the stop words and train again. As a further improvement we remove the words that occur less than 5 times across the whole set. Finally, we incorporate stemming which is sometimes found to improve recall [17, 26]. The results in the form of classification accuracy over the training set and averaged over cross-validation runs are presented in figure 3.2 (we use 10-fold cross validation as described in the previous section). With raw abstracts we get a classification accuracy of 72.29% using cross-validation. Removing the

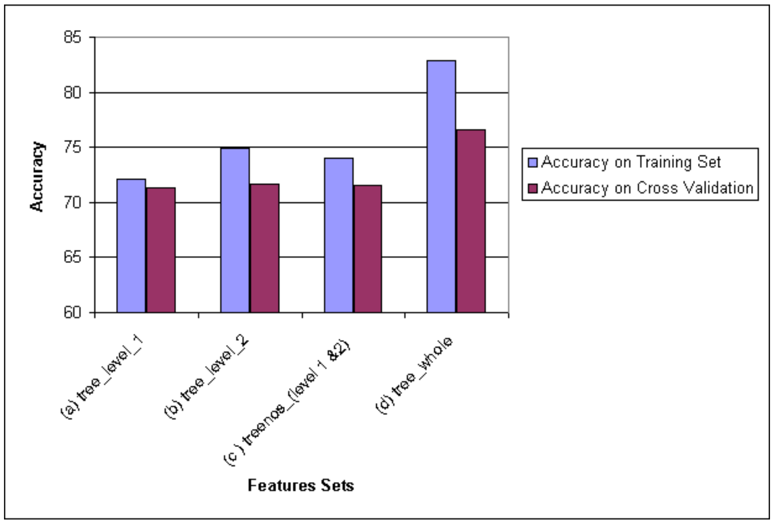

Figure 3.3: Classification accuracy with different methods of preprocessing MeSH tree labels: (a) Tree numbers from level 1 (b) Tree numbers from level 2 (c) Numbers from levels 1 and 2 (d) Complete tree numbers

stop words increases it to 72.72%, and further to 72.88% when less frequent words are also removed. Stemming gives a further improvement of 0.21% in accuracy.

We see that filtering of stop words helps improve the classification accuracy. This is expected since these words do not have any discriminative effect. Removing the words with very low frequency helps improve the performance slightly, by preventing the model from over-fitting. Stemming might be helping since it normalizes the data, where similar words with slightly different syntactic structure get grouped together. It also results in an efficiency gain. The training time for the model with stemming reduces from 49.42 seconds to 37.91 seconds and the evaluation time goes down from 125.53 seconds to 85.58 seconds.

Next, we evaluated how the MeSH tree labels perform as features. In one of the cases we considered the whole tree label as given in MeSH (e.g., *A01.456.505.420* which corresponds to *eye*, *D03.438.221.173* for *Calcimycin*). Next, we considered only the top level hierarchy for each of the tree labels (e.g., *A01*, *D03*), then the second level (e.g., *A01.456*, *D03.438*) and also a combination of the two. The comparative classification accuracy over the training data as well as averaged over the cross-validation runs is shown in figure 3.3. Using the whole tree labels gives a classification accu-

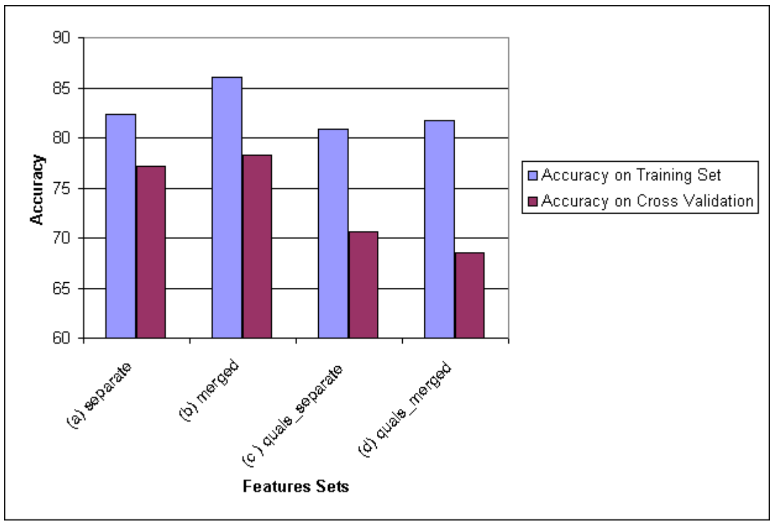

Figure 3.4: Classification accuracy with different methods of using the MeSH headings: (a) descriptors with separate word (b) decriptor words together (c) qualifier as separate words (d) qualifiers as part of descriptor

racy of 76.58% over the cross-validation set. Using only level one numbers we get 71.28%, with level two 71.63% and, using both the levels, a slightly lower:- 71.58%. It is interesting to note that generalization of the tree hierarchy (e.g., using the top or the second level) hurts the performance.

We then evaluate the different ways in which the MeSH *descriptors* perform as features. First, we use the MeSH descriptors alone, interpreting the different words in each descriptor as separate: e.g., “Heart Disease” would be interpreted as two separate words “heart” and “disease”. Next, we use the MeSH descriptors interpreting different words in the descriptors as one, e.g “Heart Disease” becomes a single word “heart_disease”. Finally, we combine the qualifier (see section 3.1.2) information to generate specific concepts for the documents together with the descriptors that appear without any qualifying term. Looking at figure 3.4, we can see that using MeSH descriptors alone, while interpreting the different words in the same term as part of the same phrase, gives the best performance. Descriptors interpreted as separate words give a classification accuracy of 77.14% over the cross-validation set. Interpreting them as phrases (which should be the more obvious choice) gives an accuracy of 78.29%, which is also the highest we achieve over all the different feature sets. A combination with a qualifier in any form decreases the accuracy substantially to around 70%.

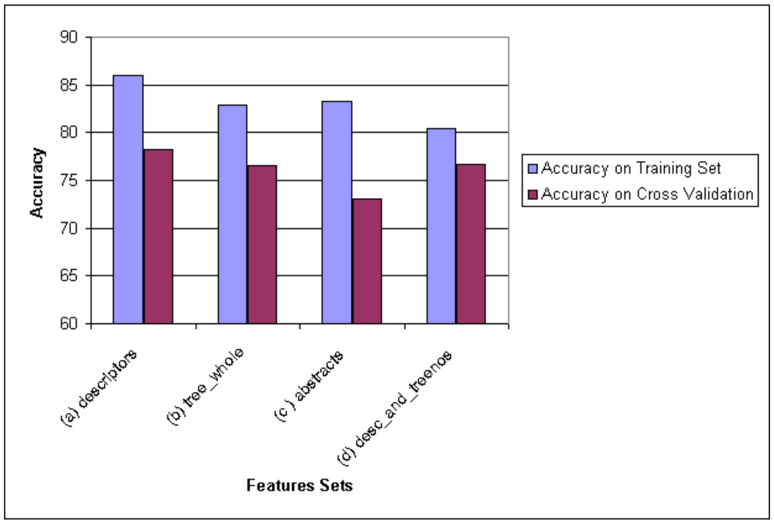

Figure 3.5: Classification accuracy for different feature sets: (a)descriptors as phrases (b) Whole tree numbers (c) Abstracts cleaned and stemmed (d) descriptors together with tree numbers

Next, we compare the classification accuracy of the different feature sets in figure 3.5. The first one is (a)MeSH descriptors, then the (b)tree labels, next the (c)abstracts with stemming and finally the (d)MeSH descriptors combined with tree labels from levels one and two. We see that the use of MeSH descriptors as features outperforms all the rest. Descriptors do better since they tend to use standardized terms and omit details like experimental setup, etc. The tree labels, though capturing the complete document information, contain some noise in the form of ambiguity due to the terms mapping against the MeSH hierarchy [34].

For each experiment, we obtain the number of true positives and true negatives averaged over the number of cross-validation sets. We use them to compute the precision and recall values for each of the cases (see figure 3.6). When only abstracts are involved, the number of false positives and false negatives are pretty much the same. Using qualifiers in any combination reduces the precision (by adding spurious matches). When using descriptors, the precision is a bit higher than recall. When combining descriptors and tree labels, the recall is much higher than precision. Figure 3.7 contains the comparison of the F-Measure obtained using different feature sets.

We observe that the classification using MeSH descriptors as features gives the best accuracy in terms of both precision and recall. We performed

Feature Set	TN	FN	TP	FP	Precision	Recall	Fmeasure
abstracts_cleaned	673	255	695	254	73.235	73.1579	73.1964
abstracts_cleaned(only_stop_words)	705	223	660	289	69.5469	74.7452	72.0524
abstracts_cleaned_stemmed	679	249	693	256	73.0242	73.5669	73.2946
descriptors	843	157	722	277	72.2723	82.1388	76.8903
desc_and_quals_with_spaces	874	126	538	461	53.8539	81.0241	64.7023
desc_and_quals	883	117	486	513	48.6486	80.597	60.6742
tree_level_1	756	244	669	330	66.967	73.2749	69.9791
tree_level_2	786	214	646	353	64.6647	75.1163	69.4997
tree_whole	796	204	735	264	73.5736	78.2748	75.8514
desc_and_tree	790	210	742	257	74.2743	77.9412	76.0636

Figure 3.6: Performance of the classifier using the different feature sets.

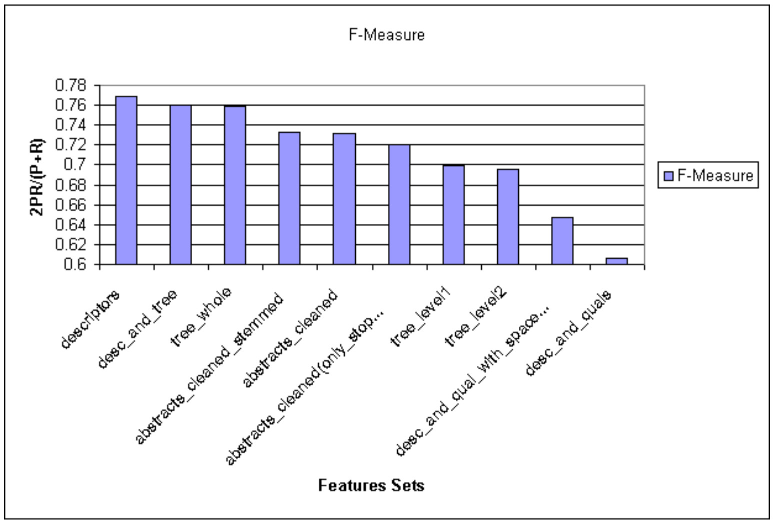

Figure 3.7: F-measure for the different feature sets.

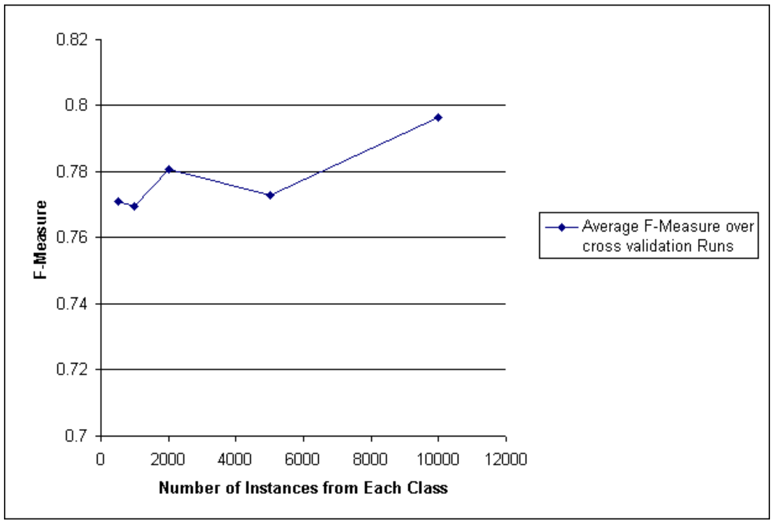

Figure 3.8: F-measure for different input sizes.

another experiment where we trained the model, using MeSH descriptors, on different sizes of training instances: we started with 500 instances of each class and trained till $10K$ instances. We compare the classifier accuracy in terms of the F-Measure in figure 3.8. We see that the accuracy of the model increases with the size of the input class instances. We think that this is because the data used from PubMed ranges over a period of 40 years, causing differences in representation styles and terminology. To test this we construct a new training set using documents that are sampled over the whole time range. We get 83.37% classification accuracy over this set using cross validation; supporting our explanation.

Chapter 4

Retrieval Module

The task for retrieval is to find all documents that are associated with a given gene. Normally a document that contains the gene name talks about some aspects of the gene (identification, mapping or function). So the task of the retrieval module is to retrieve all documents that contain the gene name. We match the gene names against the abstracts and titles of the PubMed articles since the full text is not available for most articles. It should rarely happen that an article that talks about a gene does not have the gene name in the abstract or title. Most of the time such cases occur when the article has a loose association with the gene[1]. Since we are interested in articles that talk about gene functions, we can safely ignore such cases.

The names and symbols for the genes are contained in the LocusLink database, so the retrieval should essentially be just a simple match using a text index over the PubMed articles. However there are some challenges that make the task interesting, which we discuss in the following section.

4.1 Challenges

Gene names appear in several alias forms (synonyms), e.g., the gene for LocusID-123 could be represented using "adipose differentiation-related protein" or "adipophilin". Since there is no accepted standard way of representing gene names, different authors use different forms of the same names to refer to the gene. E.g., "adrenergic receptor alpha 1d" vs. "alpha(1d)-

[1]One example of such an article is PubMedID-11076861 titled "RIKEN integrated sequence analysis (RISA) system–384-format sequencing pipeline with 384 multicapillary sequencer" which is associated with 6990 Mice genes out of the total of 58498 contained in LocusLink.

adrenergic receptor". Sometimes in the abstract we might find references like "FGF-1, -2, -4, -5, and -7" which could match any of "FGF-1", "FGF-2", "FGF-4", "FGF-5" or "FGF-7". Authors could simply use "FGF 1-7" if all the numbers are applicable. Another example of variation is the use of abbreviations; "GlyR alpha1" instead of "Glycerine Receptor alpha-1". This is one of the difficult cases to detect. The easier ones to detect could be simple things like use of different separators, e.g., "alpha-1" instead of "alpha(1)". We need to normalize and expand the gene names to capture these variations.

Names for different genes could also have large overlap. E.g., "arylamide acetylase 2" and "arylamide acetylase pseudogene". Such variations lend the use of fuzzy matches imprecise; more so for cases with higher overlap like "ATP" and "ADP" or "Agpat1" and "Agpat2".

Document search for a gene should include all the possible aliases that can be used to represent it. LocusLink contains alias names and symbols for genes but it is not exhaustive. We could use synonym information contained in SWISSPROT [7], but a study by Hong et al.[51] as part of their experiments reveals that there are many synonym pairs in SWISSPROT that domain experts disagree on (318 out of 989 pairs). We restrict our list to that obtained from LocusLink augmented with the morpho-syntactic variations that could occur in the representations. In the next section we show how we detect and generate such variations for gene names.

4.2 Identifying variations in gene names

Our goal is to discover the possible variations that can occur in gene name representations. This should be automated with minimum of human effort. We present a semi-automated technique to identify such variations. We also present a way to discover further variations that were not caught using the semi-automated procedure.

4.2.1 N-Gram Overlap

We want to find the form in which a gene is represented in articles that are known to talk about it. Through LocusLink we have a rich collection of genes and references to the articles associated with the genes. The first step in identifying patterns of variation is to locate the variant form of the gene name in the article text. We automate this step by using the N-Gram overlap measure [16]. "N-Grams" are simply n-long strings of continuous characters in a given document/string. The distribution of n-grams between pairs of strings is compared, and a score is computed that represents the similarity between them. The main idea behind using n-grams is that similar words will have a high proportion of n-grams in common.

Typical values for n are 2 or 3 corresponding to the use of digrams or trigrams, respectively. For example, the string "arylamide acetylase" results in generation of the following grams:

$$\begin{aligned} Digrams &\equiv \{ar,\ ry,\ yl,\ la,\ am,\ mi,\ de,\ e\ ,\ \ a,\ ac, \\ &\quad ce,\ et,\ ty,\ yl,\ la,\ as,\ se\} \\ Trigrams &\equiv \{ary,\ ryl,\ yla,\ lam,\ ami,\ mid,\ ide,\ de\ , \\ &\quad e\ a,\ \ ac,\ ace,\ cet,\ ety,\ tyl,\ yla,\ las,\ ase\} \end{aligned}$$

To compute the similarity of two strings using this method, we first compute the n-gram sets for both the strings and then calculate the overlap using the "Dice Coefficient"[46]. The Dice coefficient $\mathcal{D}$ for two sets A and B of sizes $|A|$ and $|B|$ is given by

$$\mathcal{D} = \frac{2|A \cap B|}{|A| + |B|} \tag{4.1}$$

This overlap measure penalizes the presence of extra characters beyond the ones common to the two strings. Thus two strings with same overlap get higher score when the non-overlapping regions are smaller in size.

We take the abstract and title of the articles associated with the genes and compute the N-gram overlap of all the possible subsequences of words against all known alias forms of the gene. We use character level digrams and trigrams with Dice Coefficient as the overlap measure. The word sequences in the abstracts/title that have high similarity to one of the known alias forms of the query gene are reported as an output of the module.

4.2.2 Inspection and Rule generation

The procedure outlined above yields high similarity pairs of strings with one of them corresponding to the known alias form of a gene name and the other to the actual variant form of representation found in article text. We process this list to remove the ones that are exactly identical (note that identical strings will receive the highest similarity coefficient of 1). Next we remove those that lie below a threshold, that is obtained using quick inspection of the list (we used a threshold of 0.5). This yields a set of original forms and their variants. Selected overlap matches are shown in figure 4.1.

Known Alias Name	**Best match variant in text**
HLA-DQB1	hla-dqb
DNA synthesis inhibitor	inhibitors of dna synthesis
phospholipase C, gamma 1	phospholipase c gamma 1
adrenergic receptor, alpha 1d	alpha 1d-adrenergic receptor
Janus kinase 2 (a protein tyrosine kinase)	protein tyrosine kynase
golgi protein, 73-kD	golgi protein
luteinizing hormone/choriogonadotropin receptor	luteinizing hormone–choriogonadotropin (lh/hcg) receptor

Figure 4.1: Some selected Overlap Pairs for gene names and their variants

A, B	$\Rightarrow$	$\begin{cases} A\ B & \textit{removal of comma} \\ B\ A & \textit{rearrangement of tokens} \end{cases}$
$\left.\begin{matrix} A([n]) \\ A\_[n] \\ A\text{-}[n] \end{matrix}\right\}$	$\Rightarrow$	$A\ [n]$ *where* $[n]$ *is the set of numerals*
$A[n]$	$\Rightarrow$	$A\ [n]$ *addition of spaces (normalization of numerals)*
$A\ [n]$	$\Rightarrow$	$A[n]$ *removal of spaces (denormalization of numerals)*
$A(B)$	$\Rightarrow$	$\begin{cases} A\ B & \textit{removal of parentheses} \\ A & \textit{removal of terms in parentheses} \end{cases}$

Figure 4.2: Rules for expanding gene names

We inspect the pairs obtained to identify the patterns of variation in gene names. These patterns are used to generate rules to transform the names to obtain a broader set of alias forms for the gene names. The rules that we generate are shown in figure 4.2. Such rules are syntactic in nature, sometimes there are variations of semantic nature that cannot be captured this way. We plan to explore semantic variations as part of our future work.

4.3 Retrieving the Documents

We use IBM's DB2 Universal Database to store PubMed documents including the abstracts, titles and other annotations. We build text indexes on these fields using DB2 Text Extender, which is then used to search for documents that contain a given set of terms. We retrieve all the documents that match one of the known alias forms of the gene or the variations created using the rules discovered above (variant forms are generated for all the aliases). We also normalize the word sequences in abstract/titles before matching against the known alias names. During normalization we replace all delimiters by spaces. E.g., “alpha(1)” and “alpha-1” become “alpha 1”. We also remove commas and sentence terminators like “.” and “?”. The retrieval score for each document is obtained by combining the frequency of occurrence of the term in the document and the relative size of the retrieved document. The exact details of this scoring function were not available to us at the time of writing this report.

We also filter out documents that do not correspond to the organism that the query gene belongs to (note that each query in the trec task consists of a gene name and a corresponding organism). This filtering is needed because similar genes from two organisms can have the same name. We describe in next section the procedure to filter out such documents.

4.3.1 Organism Filtering

Similar genes with same name can occur in multiple organisms, e.g., the gene named c-myc which stands for “cellular myelocytomatosis oncogene” can be found in different organisms including humans and chickens. In humans it is located on chromosome 8 and is involved in the pathogenesis of Burkitt’s lymphoma. In chickens, c-myc activation by avian leukosis virus appears to result in the development of lymphoid leukosis. Most of the time we are interested in documents that talk about the function of the gene corresponding to a given organism.

PubMed records do not contain an “organism annotation” for the documents. However this information can be inferred from other data present for each document, one of them is MeSH (Medical Subject Heading) descriptor. Looking at these MeSH descriptors we can identify the organism corresponding to a document. For example a document that talks about fruitfly “Drosophila melanogaster” contains the term *Drosophila melanogaster* as one of the mesh headings. The trick however is that not all the organism names are used *as is* ; e.g., the term *Human* is used instead of “Homo sapiens” which is the species name[2].

We use the combined information in LocusLink and PubMed to identify the descriptors used to characterize the organisms for PubMed documents. We collect the PubMed references (as described before, LocusLink has a set of references to PubMed documents relevant to the gene) for documents corresponding to each of the organisms[3] in LocusLink, this gives us different

[2]Note that one term is from LocusLink while the other is from MeSH.

[3]LocusLink contains genes from eight different organisms

Bos taurus	Animal **Cattle** Molecular Sequence Data
Caenorhabditis elegans	Animal **Caenorhabditis elegans** Support, Non-U.S. Gov't
Danio rerio	Animal **Zebrafish** Support, Non-U.S. Gov't
Drosophila melanogaster	Animal Support, Non-U.S. Gov't **Drosophila melanogaster**
Homo sapiens	**Human** Support, Non-U.S. Gov't Molecular Sequence Data
Human immunodeficiency virus 1	Human Support, Non-U.S. Gov't **HIV-1**
Mus musculus	Animal **Mice** Support, Non-U.S. Gov't
Rattus norvegicus	Animal **Rats** Support, Non-U.S. Gov't

Figure 4.3: Most Frequent Mesh terms for documents grouped by organism

document classes each corresponding to one organism. We then run a query against the database to find the top MeSH terms for each of the document classes created above. The top three terms for each organism are shown in figure 4.3. The MeSH terms that could possibly map to LocusLink organism names are shown in boldface.

Looking at the most frequent MeSH descriptors for each of the document classes, we can infer the term that is used to denote the organism in PubMed. We also check if the LocusLink organism name in full or part is used in PubMed for the same; e.g., the terms *Drosophila* and *Caenorhabditis* also appear in the MeSH headings contained in PubMed. None of the other organism names appear in their original LocusLink form. The terms that

Organism name from LocusLink	MeSH descriptor to look for
Bos taurus	Cattle
Caenorhabditis elegans	Caenorhabditis elegans, Caenorhabditis
Danio rerio	Zebrafish
Drosophila melanogaster	Drosophila melanogaster, Drosophila
Homo sapiens	Human
Human immunodeficiency virus 1	HIV-I
Mus musculus	Mice
Rattus norvegicus	Rats

Figure 4.4: MeSH terms used to map documents to organisms

we use in our system finally for the mapping are shown in the figure 4.4. Using these terms we map the documents in our collection to organisms. Some of the documents map to multiple organisms and a few map to none.

4.3.2 Experimental Evaluation

We use the PubMed references contained in LocusLink to evaluate our retrieval module. As described before, PubMed references in LocusLink refer to the list of PubMed documents associated with the genes. We used a subset of 50 genes from LocusLink and retrieved the documents relevant to them using our module. For measuring the recall we measure the number of known PubMed references (using full PubMed) that are retrieved.

Figure 4.5 shows the recall that was obtained using this set. We had a total 606 known PubMed references for this set. We see that using expansion and normalization results in significant improvement of recall.

PubMed references are a good collection but are in no way exhaustive (and are sometimes incorrect) making it impossible to use them for precision judgments. However since our retrieval algorithm requires exact matches of gene names and their variations, high recall is a fair indicator of performance (we do not introduce any variations that will match names of other genes). However the PubMed references were more carefully annotated (still not exhaustive but fairly accurate) for the period April 1^{st}, 2002 onwards. We create a subset of PubMed for the range 4/1/2002 to 4/1/2003 and use that to evaluate how our algorithm effects precision.

We construct a query set of 50 genes and retrieve the documents relevant to them. We also construct a *relevance set* that contains known PubMed references for each gene (we obtain 308 references from LocusLink). The

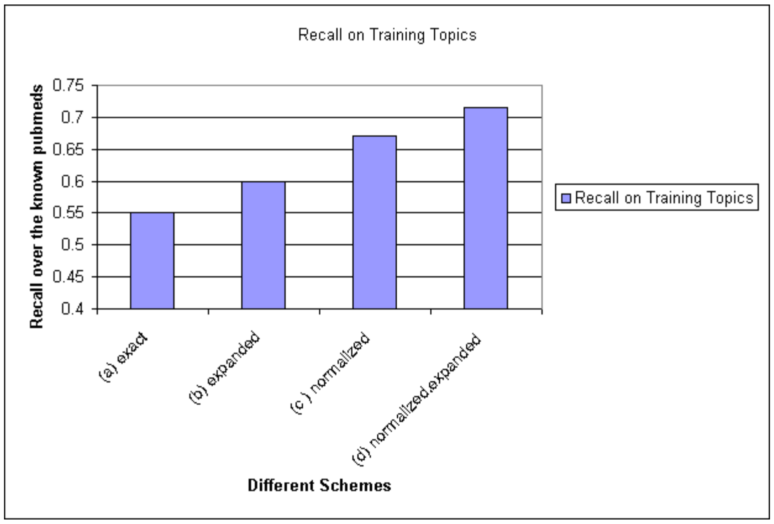

Figure 4.5: Recall with different variation schemes: (a) No variations (b) With rule-based expansions (c) With normalization (d) Both normalization and expansion

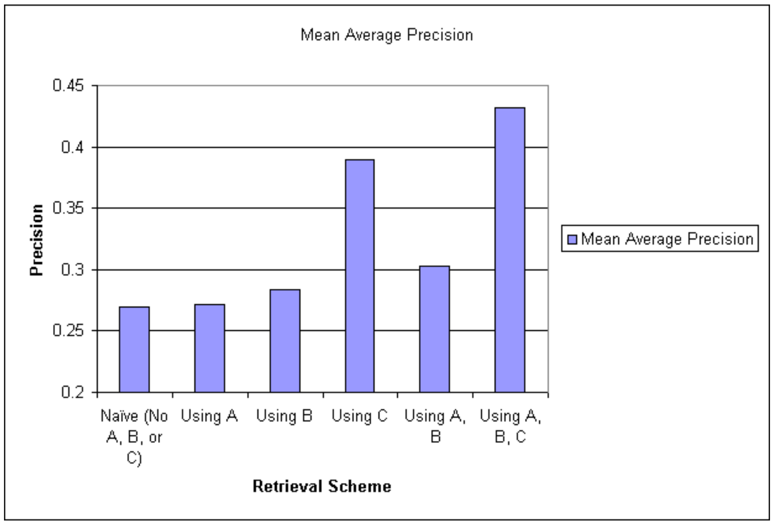

Figure 4.6: Mean Average precision with different combinations of (A) Expansion, (B) Normalization and (C) Organism Filtering

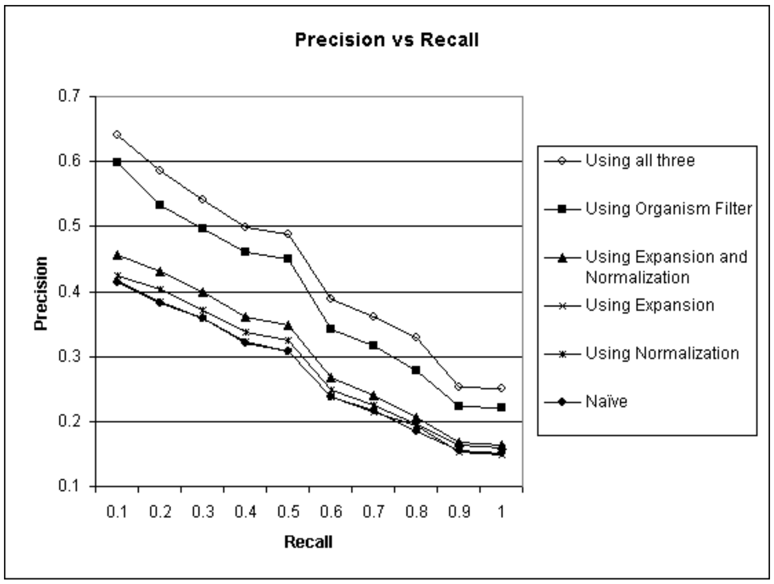

Figure 4.7: Interpolated precision vs recall for different combinations.

results of retrieval are compared against this set using TREC [8] style evaluation. Precision is computed for each query gene at different points of recall and average precision is obtained for the query. The mean of these average precision values for the individual queries is used as an indicator of overall performance. For each query (gene name) we output a set of top 1000 documents that are relevant to its function.Figure 4.6 shows the mean average precision (MAP) obtained in different cases with and without expansion, normalization and organism filtering. We also report the recall for each of the cases in figure 4.8. Organism filtering reduces the number of PubMed references that are retrieved (from the above set) but it improves the precision significantly (goes from 0.27 to 0.38). Note that organism filtering hugely reduces the total number of documents retrieved (from 89981 to 3322). Also the use of expansion and normalization helps with both precision and recall. We obtain the best precision of 0.4428 using normalization, expansion and organism filtering together. Note that the final precision is low because the set of PubMeds we use to evaluate are not exhaustive. So, we also plot the precision obtained in each scheme at different points of retrieval. Figure 4.7 shows the interpolated recall[4] at different retrieval points

[4] "Interpolated" means that, for example, precision at recall 0.10 is taken to be MAXIMUM of precision at all recall points ≥ 0.10.

Retrieval Scheme	# Relevant Retrieved	Total # Retrieved
Naive (Without A, B or C)	235	89981
Using A	247	93933
Using B	239	90279
Using C	230	3322
Using A, B	258	94338
Using A, B, C	253	4739

Figure 4.8: Recall with different combinations of (A)Expansion, (B) Normalization and (C) Organism Filtering (over a total of 308 documents).

for each scheme. Values are averaged over all queries.

4.3.3 Improving it Further

We identify a set of *false negatives* for our best system above (normalization combined with expansion). *False negatives* are those documents that appear in the PubMed references for the gene in LocusLink, but for which we could not find a gene name match in the abstract/title.

Inspecting these *false negatives* we discovered variations that were missed by our gene name overlap module. Around 10% of the missed documents were general in nature, in a sense that they were not associated with any particular gene but talked about a group of them, so the gene name did not appear in the corresponding abstracts or titles. For around 20% of the misses (5% of the total) the gene name did not appear in the abstract or the title. Of the remaining misses, some were of the kind that could be captured by our gene name expansion modules if we used a combination of the rules in sequence. E.g., the PubMed document for identifier 11580237 appears in the reference list of Locus ID 2741. The article contains “glycine receptor alpha1” instead of “glycine receptor, alpha 1 (startle disease/hyperekplexia, stiff man syndrome)”. We would have found this match if we applied our rules for removal of strings in parentheses and normalization of numerals in sequence. We added this rule combination to our expansion module and that results in 3% increase in recall.

We also find few interesting misses that would have required more intelligent matching of the semantics. E.g., at one place the authors first report the abbreviation “GlyR” for “Glycine receptor” and then use the term “GlyRalpha1” for “Glycine Receptor alpha-1”. Sometime the gene is mentioned as part of a sequence of genes, e.g., “FGF-1, -2, -4, -5, and -7” instead of “FGF7”. Lastly there were some variations that would need a semantic parse of the text, e.g., “Two cDNAs encoding variants (alpha 1 and alpha 2) of the strychnine binding subunit of the inhibitory glycine receptor (GlyR)” should match to genes “glycine receptor alpha1” and “glycine receptor alpha2”.

4.4 Combining the Modules

Once we have the classification and the retrieval modules, they can be combined to generate for a given gene a set of documents that talk about its function. The algorithm for generating such GeneRIFs is outlined in algorithm 1. The algorithm proceeds by expanding the set of aliases names for the query gene and finding the documents that contain one of the name forms. These aliases are used to retrieve documents that match them. The retrieval engine provides a score that reflects how well the document matches the query terms. Next we compute the score that the classifier assigns to the document (this is simply the probability of the document talking about gene function). The two scores are combined to get the final score that is used for producing a ranked output.

Algorithm 1 RetrieveReferencesIntoFunction(L)

```
 1: for all $l \in L$ do
 2:    $\mathcal{A}(l)$ = GetAliases($l$)
 3:    $\mathcal{A}'(l)$ = ExpandNames($\mathcal{A}(l)$)
 4:    $\mathcal{D}(l) = \phi$
 5:    for all $n \in \mathcal{A}'(l)$ do
 6:       $\mathcal{D}(l) = \mathcal{D}(l) \cup$ RetrieveMatchingDocs($n$)
 7:    $\mathcal{D}'(l)$ = FilterOrganism($\mathcal{D}(l)$)
 8:    for all $d \in \mathcal{D}'(l)$ do
 9:       $S_1(d)$ = getClassificationScore($d$)
10:       $S_2(d)$ = getRetrievalScore($d$)
11:       $S(d)$ = combineScores($S_1$, $S_2$)
12:    $\mathcal{F}(l)$ = getTopRankedDocs($\mathcal{D}'(l)$, $S$)
13: return $\mathcal{F}$
```

4.4.1 Generating a Combined Score

As described above we get two sets of scores for every document d that is retrieved. $\mathcal{S}_1(d)$ is the score from the retrieval module that tells how much confidence we have in the document being relevant to the query gene. $\mathcal{S}_2(d)$

is the score from the classification module that tells how likely the document is to be talking about gene function. We need to combine the two scores to get the final score that will be used for ranking. This is not a *"two heads better than one"* case. A document needs to get a high score from both the modules to be ranked high. We thus combine the scores using a weighted multiplicative combination. The combined score $\mathcal{S}(d)$ is given as follows

$$\mathcal{S}(d) = \mathcal{S}_1(d)^{\lambda_1} * \mathcal{S}_2(d)^{\lambda_2} \tag{4.2}$$

Where λ_1 and λ_2 are the weights. We perform retrieval with different values of these weights, the results are reported in the next section.

4.4.2 Experimental Evaluation

We use data from LocusLink to create a test set consisting of 50 different genes and associated GeneRIFs that serve as relevance documents (see chapter 3 for details on LocusLink and GeneRIFs). We use the data from the remaining genes in LocusLink to retrain the best classifier that we obtained earlier[5]. We measure the performance of the combined system using an experimental setup similar to that used for evaluating precision of the retrieval module (see section 4.3.2), but for relevance measure we use the GeneRIFs corresponding to the query genes instead of the PubMed references that were used earlier (we obtain 299 GeneRIFs from LocusLink).

We rank the retrieved documents using a combination of scores from the retrieval and classification modules as described in the previous section. We fix λ_2 (the weight assigned to the retrieval score) to 1.0 and vary λ_1 (the weight assigned to the classifier score). Figure 4.9 shows the mean average precision for different values of the weight λ_1. The precision improves marginally with very low values of λ_1, compared to the one obtained without using the classifier (using a weight of 0). The best performance is observed when λ_1 equals 0.0032. The precision goes down with higher values of the weight, for weights above 0.008 precision is lower than that obtained without using the classifier and it drops further as the weight is increased.

The classifier does not provide any significant improvement to retrieval even though we obtained good results when it was used in isolation (as reported in chapter 3). It may be because the classifier when used in isolation does not rely on the fact that there is a detailed query describing the relevant documents, rather it classifies documents as being GeneRIFs or not independent of the query. Thus, it is more generally applicable. Moreover

[5]The best classification is achieved by using MeSH descriptors for the documents.

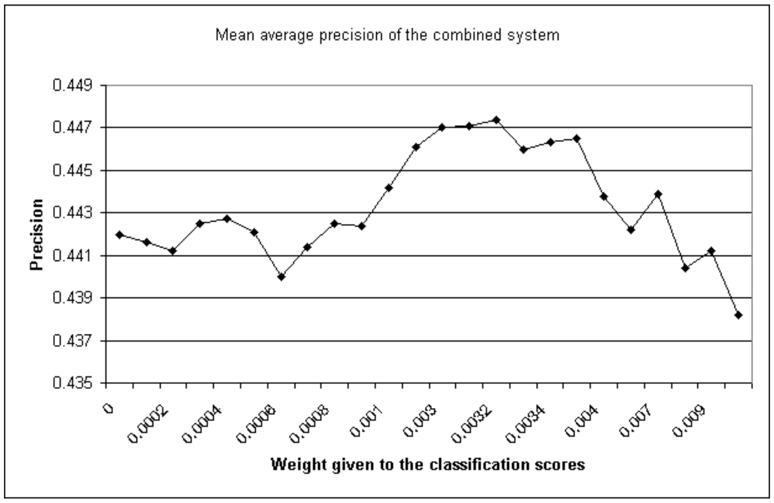

Figure 4.9: Mean Average Precision with different weights for score combination

producing a list of ranked results on a subset of documents (as done in the retrieval task) is different than making a *yes/no* decision on every individual document (as in the isolated classification task).

Chapter 5

Conclusions

We presented a system for identifying documents that talk about function of a given gene. Identifying such information is an important task for bioscience researchers who could use the system to narrow down the subset of documents and localize the information needed during the research process. Another possible use of the system could be to aid human curators of the bioscience resources, e.g., adding gene function information of the kind contained in the GeneRIF field in LocusLink database.

The main contributions of this work are (a)Classifying gene function information for documents based on domain knowledge derived from MeSH, (b)Semi-automatic rule-based expansion of gene names to identify variant forms that are used in literature, (c)a simple and effective means of identifying organism information for PubMed documents.

5.1 Future Work

The present system is an initial attempt and several extensions and improvements are possible. One such extension could be the usage of social information available for the PubMed documents, e.g., the journal in which the document appears could give important cues about its main focus etc. We would also like to experiment with other features from PubMed: e.g., list of chemicals associated with a document (available as a separate hierarchy in MeSH), semantic categories for MeSH terms, etc.

It would be also interesting to try to go to the sentence level: whether an abstract talks about some function of the gene or not, is determined locally. So, if we extract positive and negative sentences, we could train a classifier which identifies sentences that talk about gene function. Another possible future direction would be to parse the text and identify lexico-semantic features that determine the presence of function information in a piece of text. This would also enable us to identify cases when an article talks about multiple genes but the function of only one of them. Chang et al. [13] describe a computational formalism that captures structural relationships among participants in a dynamic scenario. This structured event representation using frame semantics enables them to make context sensitive inferences.

Further as we observed in chapter-4.3.2 there are many semantic variations in representation of the gene names. The retrieval performance could be improved if we do an intelligent semantic analysis of the text, e.g., using normalization of abbreviations appearing in the abstract text (see Schwartz and Hearst [41]).

Bibliography

[1] Genomics track of the 2003 text retrieval conference home page. http://medir.ohsu.edu/~genomics/.

[2] Medical subject headings. http://www.nlm.nih.gov/mesh/meshhome.html.

[3] National center for biotechnology information. http://www.ncbi.nlm.nih.gov/.

[4] National library of medicine (nlm). http://www.nlm.nih.gov/.

[5] Ncbi locuslink. http://www.ncbi.nlm.nih.gov/LocusLink/.

[6] Nlm medline/pubmed. http://www.nlm.nih.gov/pubs/factsheets/medline.html.

[7] Swiss-prot protein knowledgebase. http://us.expasy.org/sprot/.

[8] Text retrieval conference (trec) home page. http://trec.nist.gov/.

[9] Usp di - united states pharmacopeia drug information. http://www.usp.org/.

[10] Waikato environment for knowledge engineering. http://www.cs.waikato.ac.nz/ml/weka/.

[11] Blaschke, C., Andrade, M., Ouzounis, C., and Valencia, A. Automatic extraction of biological information from scientific text: protein-protein interactions. In *International Conference on Intelligent Systems for Molecular Biology:ISMB* (1999), pp. 60–67.

[12] Chang, J., Raychaudhuri, S., and Altman, R. Using biological literature improves homology search. In *Pacific Symposium on Biocomputing* (2001), pp. 374–383.

[13] Chang, N., Narayanan, S., and Petruck, M. R. Putting frames in perspective. In *International Conference on Computational Linguistics* (2002).

[14] Collier, N., Nobata, C., and Tsujii, J. Extracting the names of genes and gene products with a hidden markov model. In *18th International Conference on Computational Linguistics (COLING)* (2000), pp. 201–207.

[15] Craven, M., and Kumlien, J. Constructing biological knowledge-bases by extracting information from text sources. In *Seventh International Conference on Intelligent Systems for Molecular Biology* (Germany, 1999), pp. 77–86.

[16] Damashek, M. Gauging similarity with n-grams: Language-independent categorization of text. In *Science* (February 1995), vol. 267, pp. 843 – 848.

[17] Dawson, J. Suffix removal for word conflation. In *Bulletin of the Association for Literary & Linguistic Computing* (1974), vol. 2(3), pp. 33–46.

[18] Domingos, P., and Pazzani, M. Beyond independence: Conditions for the optimality of the simple bayesian. In *International Conference on Machine Learning, ICML* (1996).

[19] Duda, R., and Hart, P. Pattern classification and scene analysis, 1973.

[20] Fukuda, K., Tamura, A., Tsunoda, T., and Takagi, T. Toward information extraction: identifying protein names from biological papers. In *In Pac. Symp. Biocomput.* (1998), pp. 707–718.

[21] Gaizauskas, R., Demetriou, G., Artymiuk, P. J., and Willet, P. Protein structures and information extraction from biological texts: The pasta system. In *Bioinformatics* (2003), vol. 19(1), pp. 135–143.

[22] Hanisch, D., Fluck, J., Mevissen, H.-T., and Zimmer, R. Playing biology's name game: Identifying protein names in scientific texts. In *Pacific Symposium on Biocomputing* (2003), pp. 403–414.

[23] Hatzivassiloglou, V., Duboue, P., and Rzhetsky, A. Disambiguating proteins, genes, and rna in text: a machine learning approach. In *Bioinformatics* (2001), vol. 17(1), pp. S97–S106.

[24] Hearst, M. A. Untangling text data mining. In *Association for Computational Linguistics* (1999). invited paper.

[25] Hole, W., and Srinivasan, S. Discovering missed synonyms in a large concept-oriented metathesaurus. In *AMIA Symposium* (2000), pp. 354–358.

[26] Hull, D. Stemming algorithms: A case study for detailed evaluation. In *Journal of The American Society of Information Science* (1996), vol. 47(1), pp. 70–84.

[27] Humphreys, B., and Lindberg, D. The umls project: making the conceptual connection between the users and the information they need, 1993.

[28] Ian H. Witten, E. F. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[29] Kamvar, S. D., Oliver, D. E., Manning, C. D., and Altman, R. B. Inducing novel gene-drug interactions from the biomedical literature. In *Stanford University Technical Report* (2002).

[30] Kyrpides, N. Completed and ongoing genome projects. Online. Internet. Available WWW: http://wit.integratedgenomics.com/GOLD/.

[31] Leroy, G., and Chen, H. Filling preposition-based templates to capture information from medical abstracts. In *Pacific Symposium on Biocomputing* (2002), pp. 350–361.

[32] Liu, H., Johnson, S. B., and Friedman, C. Automatic resolution of ambiguous terms based on machine learning and conceptual relations in the umls. In *Journal of the American Medical Informatics Association* (2002).

[33] Mitchell, T. *Machine Learning*. McGraw Hill, 1997.

[34] Nakov, P., and Hearst, M. Category-based pseudowords. In *Companion Volume of the Proceedings of HLT-NAACL* (2003).

[35] Oliver, D. E., Rubin, D. L., Stuart, J. M., Hewett, M., Klein, T. E., and Altman, R. B. Ontology development for a pharmacogenetics knowledge base. In *Pacific Symposium on Biocomputing* (2002), vol. 7, pp. 65–76.

[36] Park, J., Kim, H., and Kim, J. Bidirectional incremental parsing for automatic pathway identification with combinatory categorial grammar. In *Pacific Symposium on Biocomputing* (2001), vol. 6, pp. 396–407.

[37] Proux, D., Rechenmann, F., Julliard, L., Pillet, V., and Jacq, B. Detecting gene symbols and names in biological texts: A first step toward pertinent information extraction. In *Genome Informatics* (1998), vol. 9, pp. 72–80.

[38] Pustejovsky, J., Castao, J., Zhang, J., and Cochran, M. K. B. Robust relational parsing over biomedical literature: Extracting inhibit relations. In *Pacific Symposium on Biocomputing* (2002), pp. 362–373.

[39] Rindflesch, T., Tanabe, L., Weinstein, J., and Hunter, L. Edgar: extraction of drugs, genes and relations from the biomedical literature. In *Pacific Symposium on Biocomputing* (2000), pp. 517–528.

[40] Rosario, B., and Hearst, M. Classifying semantic relations in bioscience texts. Submitted for publication.

[41] Schwartz, A., and Hearst, M. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of the Pacific Symposium on Biocomputing (PSB 2003), Kauai* (2003).

[42] Sekimizu, T., Park, H., and Tsujii, J. Identifying the interaction between genes and gene products based on frequently seen verbs in medline abstracts. In *Genome Informatics* (1998), vol. 9, Universal Academy Press, pp. 62–71.

[43] Shatkay, H., Edwards, S., Wilbur, W., and Boguski, M. Genes, themes, and microarrays: using information retrieval for large-scale gene analysis. In *8th Int. Conf. on Intelligent Systems for Mol. Bio. (ISMB 2000)* (2000).

[44] Swanson, D., and Smalheiser, N. An interactive system for finding complementary literatures: A stimulus to scientific discovery. In *Artificial Intelligence* (1997), vol. 91(2), pp. 183–203.

[45] Thomas, J., Milward, D., Ouzounis, C., Pulman, S., and Carroll, M. Automatic extraction of protein interactions from scientific abstracts. In *Proceedings of the Pacific Symposium on Biocomputing* (2000), pp. 541–551.

[46] van Rijsbergen, C. J. *Information Retrieval*, 2nd ed. Butterworth-Heinemann, 1979.

[47] Wilbur, L. T. W. Tagging gene and protein names in full text articles. In *Workshop on Natural Language Processing in the Biomedical Domain* (2002), pp. 9–13.

[48] Wilbur, W., and Kim, W. Flexible phrase based query handling algorithms. In *ASIST* (2001).

[49] Yakushiji, A., Tateisi, Y., Miyao, Y., and Tsujii, J. Event extraction from biomedical papers using a full parser. In *Pacific Symposium on Biocomputing* (2001), pp. 408–419.

[50] Yang, Y., and Pedersen, J. A comparative study on feature selection in text categorization. In *Proceedings of the Fourtheenth Intenational Conference on Machine Learning* (1997).

[51] Yu, H., and Agichtein, E. Extracting synonymous gene and protein terms from biological literature. In *Bioinformatics* (2003), vol. 19, pp. i340–i349.