University of Alberta

TERM GENERALIZATION AND SYNONYM RESOLUTION FOR BIOLOGICAL ABSTRACTS: USING THE GENE ONTOLOGY FOR SUBCELLULAR LOCALIZATION PREDICTION

by

Alona Fyshe 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Spring 2007

Your file Votre référence
ISBN: 978-0-494-29958-6
Our file Notre référence
ISBN: 978-0-494-29958-6

This work is dedicated to my parents, who funded my undergraduate career.
Their support allowed me to focus on my studies so that my undergraduate performance could,
in turn, fund my graduate career.

Abstract

The field of molecular biology is growing quickly and research findings are being deposited into public databases. Swiss-Prot is a public database of protein sequences with annotations. Annotations in Swiss-Prot include the subcellular localization of the protein and links to literature relevant to the protein. Many of the over 200,000 protein entries in Swiss-Prot (release 49.1) lack annotations such as subcellular localization, but the majority have references to journal abstracts describing related research. These abstracts represent information that could be used to automatically generate annotations for proteins that have been studied, but remain un-annotated. Training text classifiers on abstracts is one way to generate annotations. This dissertation presents a method for generating additional text features using the knowledge represented in a biological concept hierarchy (the Gene Ontology). The structure of the ontology and the synonyms recorded in it are leveraged to significantly improve the F-measure of some subcellular localization text classifiers.

Acknowledgements

I would like to thank my supervisor, Dr. Duane Szafron for introducing me to the wonderful world of research while I was an undergraduate student. His enthusiasm has been an inspiration.

In addition would like to thank Greg Kondrak, Colin Cherry, Shane Bergsma and the whole NLP group at the University of Alberta for their helpful feedback and guidance. I also wish to thank Paul Lu, Russell Greiner, Kurt McMillan and the rest of the Proteome Analyst team.

Table of Contents

1 Introduction: Automatic Annotation **1**
- 1.1 Biological Background . . . 2
- 1.2 Biological Databases . . . 8
  - 1.2.1 Swiss-Prot . . . 8
  - 1.2.2 Trembl . . . 9
- 1.3 Controlled Vocabularies . . . 9
  - 1.3.1 The Gene Ontology . . . 9
  - 1.3.2 Other Ontologies . . . 10
- 1.4 The Annotation Bottleneck . . . 10

2 Introduction to Machine Learning **12**
- 2.1 Machine Learning . . . 12
  - 2.1.1 Evaluation . . . 13
  - 2.1.2 Machine Learning Algorithms . . . 16
- 2.2 Machine Learning in Bioinformatics . . . 20
  - 2.2.1 Sequence-Based Prediction . . . 20
  - 2.2.2 Homology-Based Prediction . . . 25
  - 2.2.3 Combinations of Homology and Sequence . . . 26
- 2.3 Conclusion . . . 27

3 Introduction: Natural Language Processing **28**
- 3.1 Areas of text processing . . . 28
  - 3.1.1 Document classification . . . 28
  - 3.1.2 Named Entity Recognition . . . 32
- 3.2 Conclusion . . . 33

4 Methods **34**
- 4.1 Gathering Data . . . 34
- 4.2 Abstract Retrieval and Preprocessing . . . 37
  - 4.2.1 Preprocessing Text . . . 38
- 4.3 Leveraging the Gene Ontology . . . 38
  - 4.3.1 Baseline . . . 38
  - 4.3.2 Synonym Resolution . . . 40
  - 4.3.3 Term Generalization . . . 40
- 4.4 Evaluation . . . 41

5 Results **42**
- 5.1 The PA Data Set . . . 42
  - 5.1.1 TFIDF . . . 42
  - 5.1.2 Redundancy . . . 43
  - 5.1.3 TFIDF vs. Redundancy . . . 50
- 5.2 Case Study . . . 51
  - 5.2.1 Example: TFIDF and a protein from the lysosome . . . 51
  - 5.2.2 Example: Redundancy and a protein from the nucleus . . . 53
- 5.3 MultiLoc Data Set . . . 56
  - 5.3.1 TFIDF . . . 56
  - 5.3.2 Redundancy . . . 57
  - 5.3.3 TFIDF vs. Redundancy . . . 60
  - 5.3.4 Comparison to MultiLoc's classifiers . . . 60
- 5.4 Discussion . . . 62

6 Conclusion **63**
6.1 Discussion of Results 63
6.2 Future Work 64
6.2.1 Incorporating Text Classification into PA's classifiers 64
6.2.2 Including Other Term Hierarchies 64
6.2.3 Using an NER system 65
6.2.4 Investigating Other Classifications 65
6.3 Summary 65

Bibliography **67**

List of Tables

2.1 The layout of a confusion matrix. 15
2.2 An example confusion matrix with high precision and low recall. 15
2.3 An example confusion matrix with high recall and low precision. 15
2.4 Probability of a particular number being rolled with dice at the "Occasionally Dishonest Casino", given the state of the dice. 19

4.1 A mapping of phrases to subcellular localizations for the nucleus localization label. 36
4.2 A summary of class distribution of the PA Data Set. 37
4.3 The average number of features per training instance for animal subcellular localization categories. 41

5.1 Precision, Recall and F-measures for three Data Sets created using the PA training set and un-normalized TFIDF. 45
5.2 Precision, Recall and F-measures for three Data Sets created using the PA training set and L_2 normalized TFIDF. 46
5.3 Precision, Recall and F-measures for three Data Sets created using the PA training set and un-normalized redundancy. 47
5.4 Precision, Recall and F-measures for three Data Sets created using the PA training set and L_2 normalized redundancy. 48
5.5 Precision, Recall and F-measures for two reduced classes of the PA training set using un-normalized redundancy. 49
5.6 Comparison of TFIDF and redundancy as importance weight measures for subcellular classification of the PA Set. 50
5.7 The top five most heavily weighted words from the abstract related to protein P11117 using Data Set 1. 52
5.8 The top five most heavily weighted words from the abstract related to protein P11117 using Data Set 3. 52
5.9 The top five most heavily weighted words from abstracts related to protein O95707 using Data Set 1. 55
5.10 The top five most heavily weighted words from abstracts related to protein O95707 using Data Set 3. 55
5.11 Summary of distribution of proteins and abstracts in the MultiLoc Data Set. 56
5.12 Precision, Recall and F-measures for three Data Sets created using the MultiLoc data set and un-normalized TFIDF. 58
5.13 Precision, Recall and F-measures for three Data Sets created using the MultiLoc data set and un-normalized redundancy. 59
5.14 Comparison of TFIDF and redundancy as importance weight measures for subcellular classification of the MultiLoc Set. 60
5.15 Comparison of MulitLoc's techniques and the techniques presented in this dissertation on the MultiLoc Data Set. 61

List of Figures

1.1 A drawing of the cells that can be seen in a thin slice of cork [24]. 2
1.2 A phospholipid molecule 5
1.3 A cross section of a phospholipid bilayer. 6
1.4 The membrane structure of Gram-negative and Gram-positive cells. 6
1.5 A sub-graph of the GO biological process hierarchy. 10

2.1 The classic supervised learning example of playing tennis, graphed in two dimensional space. 17
2.2 A diagram representing the transition of dice states at the "Occasionally Dishonest Casino" 18
2.3 A multiple sequence alignment of the beta chain hemoglobin molecule of several organisms. 22
2.4 A phylogenetic tree representing the alignment of the beta chain hemoglobin molecule of several organisms. 23

4.1 The workflow used to create data sets used in this paper. 35
4.2 An example illustrating the creation of training instances using TFIDF. 39
4.3 A sentence illustrating the three methods of abstract processing. 39
4.4 A sub-graph of the GO biological process hierarchy. 40

Chapter 1

Introduction: Automatic Annotation

"If I have seen further, it is by standing on the shoulders of giants."

-Sir Isaac Newton, letter to Robert Hooke, 1676

Biology, the study of living things, has experienced exponential growth in the last century. This growth has been expedited as biologists from different cultures and different countries come together to work on research problems. From agricultural advancement to vaccines, the field of biology has changed the way we live our lives. The biological research community is a collective. Biologists work together, and their collaboration has enabled huge advances.

This remarkable progress has ushered in new challenges, including the deluge of information that comes with such rapid growth. If a researcher cannot easily find recent and relevant research in a field, the very seeds of collaboration fail to be sown. For example, if a team of researchers is working on a disease that affects the mitochondria of a cell, they may wish to identify every protein in an organism that localizes (i.e. performs its function) in the mitochondria. A simple search for the term mitochondria in a biological journal database will turn up many articles, but some research will not be found because the articles discuss not the mitochondrion itself, but some process that takes place within the mitochondria. There is a need, in the midst of this research revolution, to gather forces and build new mechanisms for sorting information, so that it can become knowledge. The techniques covered in this dissertation address the need for automatic organization for a particular type of biological annotation called subcellular localization: where a protein performs its function in the biological cell.

In order to understand the complexity of this problem, and the resources that have been developed to tackle it, a brief introduction to biology and machine learning is necessary. This Chapter contains an introduction to molecular biology, cell biology, and a primer to a subset of topics in the exciting mix of biology and computing science, called bioinformatics. Chapter 2 is an introduction to an area of computing science called machine learning and Chapter 3 introduces an area of computing science called natural language processing. My research uses techniques from each of these areas to address the problem of information overload in the biological domain.

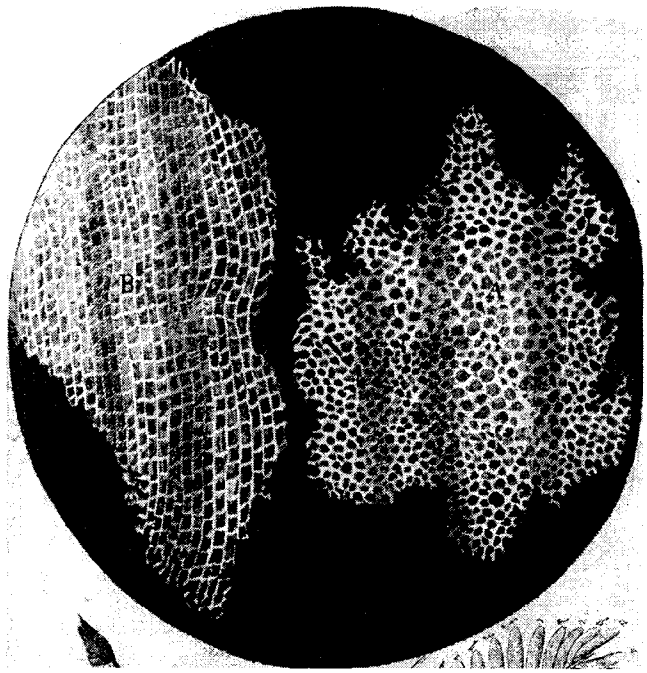

Figure 1.1: A drawing of the cells that can be seen in a thin slice of cork [24].

1.1 Biological Background

Robert Hooke was The Royal Society of London's Curator of Experiments for 15 years, beginning in 1662 [4]. During that time Hooke published "Micrographia", a book detailing the observations he made using a compound microscope. This book includes a magnified view of cork which he described as having "pores" or "cells" shown in Figure 1.1. In 1678 Hooke was asked by the Royal Society to confirm claims made in a letter from Antony van Leeuwenhoek. The letter described "little animals" which Leeuwenhoek discovered while peering through a magnifying glass. Hooke noted the same microscopic creatures. Hooke was first to note that organisms visible to the naked eye are made up of tiny biological units, called cells, and Leeuwenhoek was the first to observe unicellular organisms: bacteria and protozoa. With these two discoveries the field of cell biology emerged. In 1839, Matthias Schleiden and Theodor Schwann formalized cell theory, which states that all organisms are made up of one or more cells. Schwann noted in his subsequent book [37] the duality of these microscopic living entities, biological cells, which can live separately and self-sufficiently, or come together and co-operate to create a larger living entity.

What, exactly, is a cell? A cell is the biological unit of which all life is composed [47]. Every cell involves some sort of membrane that separates its internal workings from its surroundings. The specifics of what lies within the cell and what makes up the separating membrane are what differentiates cell types - variation abounds, even on the microscopic level. Even within the same organism, cells can differ greatly. From the ciliated cells of our lung tissue to the strong elongated

cells of our muscle, the cell is amazingly diverse.

Scientists have long been fascinated by both the likenesses between cells and their variances. What is it about an organism that causes its offspring to be so like itself? And what is it about the cells in specific tissue types that causes them to be different than the cells of neighbouring tissues? What causes these simultaneous similarities and differences in the natural world?

These observable differences in cells, called phenotypes, are the direct result of the genetic makeup of a cell. Scientists explored how the genes of a cell control the organism's phenotypes through a series of discoveries spanning more than a century. Many have heard the story of Gregor Mendel and his pea plants. A monk living in a monastery in what was is now part of the Czech Republic, Mendel studied the common garden pea. He noted that his pea plants had different physical characteristics, or phenotypes. Some plants were tall, some were short, some had green pea pods, and some had yellow. He wondered about this, and began to experiment with the plants by selectively breeding them and observing the dispersal of their characteristics among the offspring. He noted that when he crossed yellow-pod plants with green-pod plants, the offspring had only green pods. If Mendel allowed this second generation of green pod plants to interbreed, the next generation had some plants with green pods, but some of the yellow-pod plants reappeared. In the absence of any knowledge of what DNA (deoxyribonucleic acid) is or how it works, Mendel created an entire theory of genetic mixing based on what he called "factors". He postulated that there was some invisible entity that was passed from parent to offspring that controlled phenotype, a concept that we refer to now as "gene".

Still, scientists at the time did not know what a gene was, or how it was passed from one generation to the next. In fact, at the turn of the 20th century there was much research being done to uncover the biological basis of this phenomenon. Thanks to Robert Hooke and his popularization of the microscope, biologists were able to view chromosomes, complexes of tightly coiled DNA and protein as they replicated in cells. They watched with amazement as cells copied and then evenly dispersed their chromosomes during cell replication. This observation made them quite sure that these little structures somehow contained Mendel's "factors". But how did DNA and protein encode the phenotype of an organism? Even more contentious at the time, which encoded the phenotype? Was it DNA, protein or both? At the time, DNA seemed a very unlikely candidate. DNA is made up of only 4 types of molecules, called nucleotides, In contrast, proteins are composed of 20 kinds of molecules, called amino acids. Basic combinatorics will tell you that, given the same amount of space, you can encode more information using a 20 letter alphabet than with a 4 letter one. Given the complexity of life, biologists were sure that proteins were the means with which cells transmit their genetic material.

In 1928, it was discovered by Fred Griffith that the "factors" found in chromosomes could be transferred between bacteria [19]. His experiment involved two strains of *Pneumococcus*: a deadly strain with smooth-looking colonies, and a harmless strain that produced rough-looking colonies.

Griffith found, unsurprisingly, that heating the smooth strain, thus killing the bacterial cells, rendered the virus incapable of killing a mouse. However, injecting the same heat-treated smooth strain virus into a mouse that was also infected with the non-lethal rough strain rendered the animal dead. Griffith was shocked. He performed his experiment several times and each time found the same result. Somehow, the dead smooth strain was transferring its "factor" to the non-virulent rough strain, and making it lethal. This set the stage for experiments to determine which molecule it was that encoded the factor that allowed the virulent strain to transform the non-virulent strain.

Griffith's team would not be the ones to discover the molecule that encoded the gene. It was Avery, McLeod and McCarty who extended Griffith's work and devised a way of telling which molecule of the bacterial cell was responsible for the transformation of *Pneumococcus* [5]. The biological cell is composed of 5 basic molecular components: DNA, RNA, polysaccharides, proteins and lipids. Avery et al. introduced into the heat-treated smooth *Pneumococcus* a set of enzymes that could destroy four of the five components, but not DNA. This mixture was still able to transform the rough strain and make it virulent. However, mixing the heat-treated smooth *Pneumococcus* with only DNAse, an enzyme that destroys DNA, prevented the "factor" from making the rough-strain virulent. This experiment was the first to show that DNA was the mechanism of heredity in bacteria, and subsequent experiments showed the mechanism to be the same in other organisms.

This brings us to what biologists have termed the "Central Dogma" of biology. It states that DNA (found on chromosomes) is used to make RNA (ribonucleic acid) via a process called transcription. RNA is, in turn, used to make proteins via a process called translation. Proteins are ultimately responsible for many of the actions that we perceive as phenotypes, and which Mendel studied when he performed crosses with his pea plants.

Because proteins are the basic entities that control our growth and development, biologists are very interested in their characteristics and how they behave in our cells. If we can understand how our cells work on a microscopic level we can better understand the ailments that affect us on a macroscopic level. Biologists may devote their entire lives to studying one gene, and its protein product. They may wish to understand the 3-dimensional structure of the protein, determine the proteins molecular function or where in the cell a protein performs its function. Because this study deals with the subcellular localization annotations of proteins, it is necessary to understand the structure of biological cells.

One of the simplest living cells is that of Gram-positive bacteria, named for the strong purple hue it assumes when exposed to Gram-stain dye. This propensity for binding the dye is due to the cell's wall, which contains a thick layer of sugar-based peptidoglycan molecules. These thick layers trap the molecules of the Gram stain and give the cells a characteristic purple colour after treatment. The cell's wall acts as a barrier to the outside world, selectively allowing some molecules to enter, and barring others. Beneath the cell wall is a phospholipid bilayer. A phospholipid bilayer is made of two sheets of phospholipid molecules packed closely together. A phospholipid is shaped

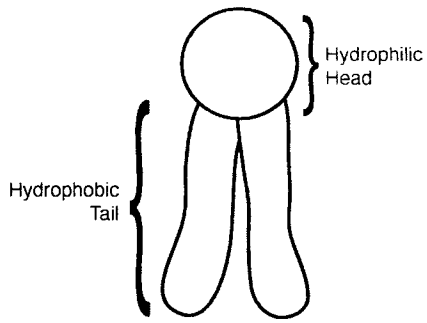

Figure 1.2: A phospholipid molecule. The head of the phospholipid, represented here with a circle is hydrophilic, meaning it is attracted to water. The tails of the phospholipid are hydrophobic, meaning they repel water.

---

like an old-fashioned wooden clothes-pin (Figure 1.2), with a hydrophilic (water-loving) head and two hydrophobic (water-fearing) tails. Because half of the molecule tends to avoid water, and half is attracted to water, when phospholipids are immersed in water (as in biological cells) they form structures that expose the hydrophilic regions and hide the hydrophobic regions. One such structure is the bilayer (Figure 1.3), in which phospholipids assemble themselves into two sheets. The bottom sheet is made of phospholipids with their tails pointing upwards, and the top sheet has phospholipids with their tails pointing downwards. This exposes the hydrophilic heads of the phospholipids to the water in the surrounding environment, and the tails are tucked away, concealed from the aqueous exterior. Within the plasma membrane of a biological cell is a liquid called cytoplasm. Dispersed in the cytoplasm of a bacterial cell are ribosomes, the machinery cells use to create protein and a small tightly coiled mass of DNA called a nucleoid. The simplicity of the Gram-positive cell means that most biologists agree upon a simple three class categorization system for this cell type: cytoplasmic (any protein performing its function in the cell), extracellular (any protein that performs its function outside of the cell, as in secreted proteins) and plasma membrane proteins (any protein involved in the structure of the membrane) (Figure 1.4).

The cell wall of Gram-negative bacteria (Figure 1.4) contains very little peptidoglycan, so it does not take on a purple colour after a Gram stain. The Gram-negative cell wall is comprised of two phospholipid bilayers: the interior bilayer (called the plasma membrane) and the exterior layer (the outer membrane). The outer membrane contains lipopolysaccharides that contribute to the cell's pathogenic properties by protecting it from the immune systems of the hosts that it invades. The space between the two lipid bilayers is called the periplasmic space. Like Gram-positive cells, Gram-negative bacteria have ribosomes and a nucleoid floating in the cytoplasm contained by the plasma membrane. The Gram-negative subcellular localization categories are: cytoplasmic (proteins that perform their function within the cell's inner membrane), inner membrane (proteins that perform their function as part of the cell's innermost membrane), periplasm (proteins that perform their

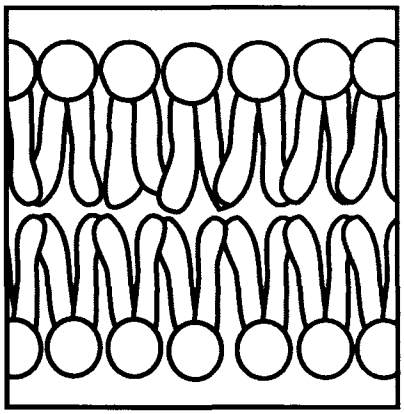

Figure 1.3: A cross section of a phospholipid bilayer. Phospholipid molecules arrange themselves so that their hydrophilic heads are exposed to the surrounding environment and their hydrophobic tails are concealed in the interior of the bilayer.

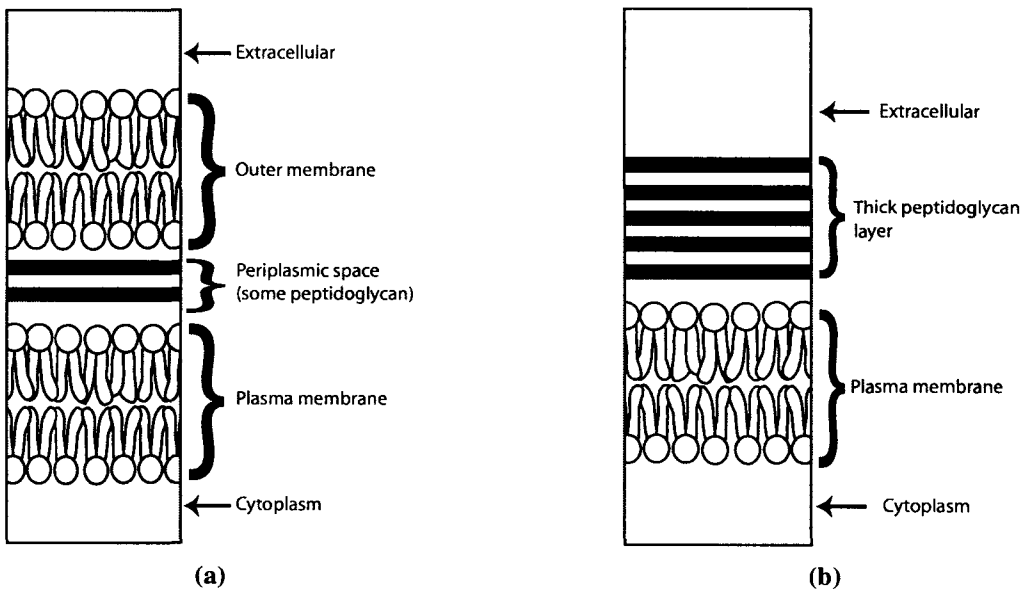

Figure 1.4: The membrane structure of Gram-negative and Gram-positive cells. **(a)** Gram-negative cells have two phospholipid bilayers that create a periplasmic space. **(b)** Gram-positive cells have a thick layer of peptidoglycan and a single phospholipid bilayer.

function in between the cells double bilayer), outer membrane (proteins that perform their function as part of the cell's outermost membrane) and extracellular (proteins that perform their function outside of the cell).

Gram-negative and Gram-positive bacteria are both prokaryotic cells, which is a class of cells where the DNA floats around freely in the cytoplasm. In the cells of eukaryotes, including humans, DNA is contained in a membrane bounded organelle called a nucleus that separates the DNA from the surrounding cytoplasm. Because eukaryotic cells are more complex they have more subcellular localizations. They have the same cytoplasmic, extracellular and plasma membrane classes as Gram-positive bacteria, but also include more specialized cytoplasmic organelles:

- The mitochondrion, the powerhouse of the cell, uses energy stored in the bonds of organic molecules to create ATP (adenosine 5'-triphosphate) through oxidative phosphorylation. In simple terms, this is the process of turning food into a form of energy that cells can readily use.

- Chloroplasts are specific to photosynthesizing organisms: plants and algae. Chloroplasts use light energy, water and carbon dioxide to create ATP and food (glucose).

- The endoplasmic reticulum (ER) is a large system formed from a folded membrane. The ER is divided into two parts, the smooth and the rough ER. The rough ER (RER) is rough because it is sprinkled with ribosomes, the cell's mechanism for creating protein. The RER is the cell's site for manufacturing, processing and storing protein. The smooth ER (SER) creates, processes and stores fats, carbohydrates and other necessary cellular molecules, aside from protein.

- The Golgi complex is created by the continual fusing of protein-containing membrane-bounded vesicles which originate from the RER. While travelling through the Golgi, proteins are processed and modified before being sent to their final destinations.

- Peroxisomes are often found in liver and kidney cells and contain the enzymes needed to carry out the waste breakdown that occurs in those organs.

- A Vacuole is a catch-all term used to describe a class of membrane-bound pockets that store and transport various molecules. For example, digestive vacuoles are created by phagocytosis, the process by which a cell envelops and "eats" molecules surrounding it. In addition, vacuoles can also be used to export or secrete molecules from within the cell, or to simply hold water to create rigidity for the cell.

- Lysosomes contain potent hydrolytic enzymes that can be used to break down the contents of digestive vacuoles, or in autophagy in which old organelles or are destroyed because they have reached the end of their usefulness to an organism.

Eukaryotic cells differ between forms of organisms. For example, only plant cells have chloroplasts. Biologists create groups for organisms based on the similarity of their cells, and thus their possible subcellular localizations. Animal cells have proteins that localize to the mitochondrion, nucleus, endoplasmic reticulum, extracellular, cytoplasm, plasma membrane, golgi, lysosome and peroxisome. Plant cells have mitochondrion, chloroplast, nucleus, endoplasmic reticulum, extracellular, cytoplasm, plasma membrane, golgi, peroxisome and vacuole localizations. Fungi cells have mitochondrion, nucleus, endoplasmic reticulum, extracellular, cytoplasm, plasma membrane, golgi, peroxisome and vacuole localizations.

With so many biologists working all over the globe, many are working on very similar problems. Often more than one team of biologists may be working on the same protein at the same time, or on similar proteins in different organisms (orthologues), and they share what they learn through journal publications. In the past two decades, the expansion of biological research has meant that most biologists cannot keep up with the publication rate even just within their particular branch of biology. This brought about the birth of the biological database.

1.2 Biological Databases

The biological research explosion made it clear that researchers had to organize their findings. If results were not easy to find in the huge amount of literature being generated then a portion of the community's work would be overlooked and become lost. This new need was filled by biological databases of genes/proteins. Biological databases are extensive databases where each entry corresponds to a gene/protein. Entries are annotated with many different facts about the matching protein, collected by researchers. Each protein entry may also be linked to the journal abstracts that describe the research pertaining to that protein. This level of organization makes it easier for a biologist to find related research. They can query the databases with keywords, and search for proteins that have a similar amino acid sequence to the protein they study in their labs.

Databases of DNA also exist, but these databases tend to contain information pertinent to a stretch of DNA before it becomes a gene, such as intron and exon splice sites. I have chosen, in this study, to use protein databases because the annotations found in protein databases are more closely related to phenotype and thus are often of greater interest to biologists. In addition, Proteome Analyst [43], the test bed for this study, uses a biological database of proteins called Swiss-Prot.

1.2.1 Swiss-Prot

Swiss-Prot [42] is a highly curated database of proteins. Curation means that a human annotates each entry of the database by hand. DNA data is deposited into EMBL's (European Molecular Biology Laboratory) sequence repository by large-scale DNA sequencing projects. From there, it is translated into amino acid sequences (proteins) and placed in Trembl (discussed in Section 1.2.2) while it awaits thorough review and annotation by Swiss-Prot's team of annotators. Because of this,

the amounts of quality annotations are higher in Swiss-Prot than in any other biological database. However, also due to Swiss-Prot's annotation requirements, Swiss-Prot's growth lags behind that of computer annotated databases. Version 50.3 of the Swiss-Prot database (July 11, 2006) contains 228,670 protein sequences from almost 10,000 different species. Swiss-Prot staff work to annotate these entries with fields such as "function", "subcellular localization", "tissue specificity" (if the protein is expressed only or to a greater extent in a particular organ or tissue type) and "interaction" (other proteins this protein interacts with, and the nature of that interaction).

1.2.2 Trembl

Trembl is a middle ground between DNA sequencing projects and protein annotation. It is "a computer-annotated supplement of Swiss-Prot that contains all the translations of EMBL nucleotide sequence entries not yet integrated in Swiss-Prot" [42]. Thus, Trembl is much larger than Swiss-Prot, containing over 3,000,000 protein sequences (version 33.3, July 11, 2006), and has grown by about 600,000 sequences since January 2006. To contrast, 4% of the entries have subcellular localization information compared to 54% in Swiss-Prot. Although there are a huge number of Gene Ontology annotations (1.8 million entries annotated) in Trembl, very few of them (0.008%) have high quality evidence codes like TAS (traceable author statement) or IDA (inferred from direct assay). Over 30% of the proteins in Swiss-Prot have GO annotations, and over 50% of those GO annotations have high quality evidence codes.

1.3 Controlled Vocabularies

Biologists have another hurdle to conquer in the quest to organize their research. The language biologists use to describe their work is very technical and displays a high level of synonymy. Synonymy (having many synonyms) means that several biologists may report their findings in journal articles, each using different terms to describe the same concept. A researcher reading these articles may not even realize that the research is related, and that they are discussing the same phenomenon. A controlled vocabulary is needed to help biologists deal with the detailed and technical language they use to describe complex biological phenomena.

1.3.1 The Gene Ontology

Recognizing the problem of synonymy, a group of biologists joined to form the Gene Ontology Consortium. The Consortium was tasked with building a term hierarchy called the Gene Ontology (GO). The GO is a directed acyclic graph (DAG) that expresses the relationships between terms (nodes or vertexes of a graph) with connections (edges of the graph) that represent parent/child relationships (see Figure 1.5). Each arrow points to the more specific term. For example, in Figure 1.5, homeostasis is a child of physiological process, which means that homeostasis "is a" physiological process. The GO also encodes synonyms in its structure, shown with grey rectangles in Figure 1.5 -

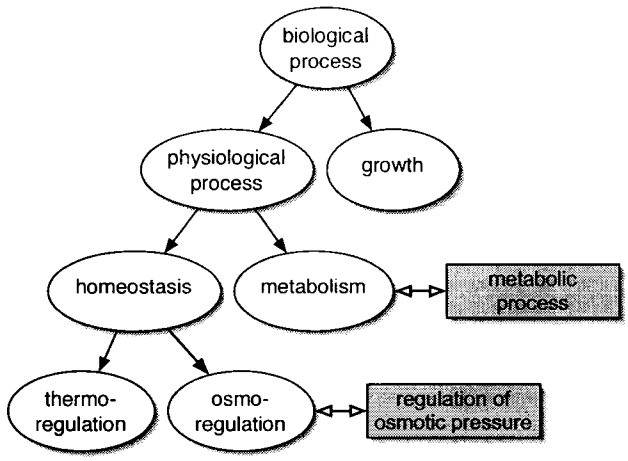

Figure 1.5: A sub-graph of the GO biological process hierarchy. GO nodes are shown as ovals, synonyms appear as grey rectangles.

---

the term "osmoregulation" has the synonym "regulation of osmotic pressure"; if two biologists each use one of these two terms to describe the same phenomenon, another biologist can use the GO to determine that they are actually discussing the same concept.

1.3.2 Other Ontologies

Aside from the GO, there are other biological ontologies. MeSH (Medical Subject Headings) is a hierarchy of medical terms that are often used as keywords for abstracts in PubMed. The enzyme class system is a hierarchical organization of enzymes. These ontologies could be substituted for, or used in conjunction with, the GO hierarchy for this study. For simplicity, and to determine the usefulness of the GO hierarchy independent of other ontologies, for this study only the GO hierarchy was used.

Biologists realize the power encoded in a controlled vocabulary, and have begun to annotate proteins in the Swiss-Prot database with GO terms. As well, the European Bioinformatics Institute maintains a mapping of GO terms to Swiss-Prot and Trembl proteins. While biological databases bring a huge amount of knowledge into one repository, controlled vocabularies, like the GO, provide the language necessary to effectively search through and interpret data.

1.4 The Annotation Bottleneck

Although annotations are extremely useful to biologists, not every protein in Swiss-Prot is annotated with every field. As stated previously, only 54% of proteins in Swiss-Prot version 50.3 have subcellular localization information and only 50% have GO annotations. However, these rates of annotation are slowly improving, which is a great accomplishment if one takes into account Swiss-

Prot's rapid growth. Swiss-Prot has added 25,000 sequences since the beginning of 2006, and has doubled in size in the last 4 years. Because annotations are so important, and because databases are growing at a brisk pace, it is imperative that annotators turn to other methods of curation to increase the speed of annotation. Using Machine Learning in conjunction with the annotated records in biological databases and term hierarchies can improve the accuracy of computer-generated annotations.

Chapter 2

Introduction to Machine Learning

The advent of computer-assisted or computer-generated annotations is a promising advance in protein annotation. Swiss-Prot curators have created a huge database of proteins with annotations, and the information represented in the repository can be leveraged to create annotations automatically for unannotated proteins. This approach is based on the culmination of decades of human annotation building on itself. The wealth of knowledge we have gathered and the extent to which we have gone to organize the findings is paying off. The annotation pipeline allows human-generated annotations to serve as fuel for computer-generated annotation systems.

How can computers be used to generate annotations? This question is being explored by an area of computing science known as machine learning. Machine learning has garnered a lot of attention in recent years as it seeks to exploit the mathematical patterns in large amounts of data that are too complicated for humans to analyze. While the name sounds fantastic and worthy of a screenplay or two[1] there is no magic; machine learning has a sound basis in statistics and mathematics. This is unfortunate for those of us who want computers to do our homework, but comforting for those who have nightmares about robots taking over the world.

2.1 Machine Learning

Machine learning starts with a set of data that is understood to be generated by some underlying process [1]. Then, through a variety of statistical methods, machine learning builds a model that represents the process that generated the data. Machine learning techniques can be divided into two groups: supervised and unsupervised. Supervised learning is a situation in which we are given a set of data points, characteristics of those data points (features) and the category or class label corresponding to each point. Formally, each training instance is a pair:

$$\mathcal{X} = \vec{x}, r$$

[1] Not to mention its ability to impress new acquaintances at dinner parties.

where $\vec{x}$ is the vector of features of a training point, and r is the class label of the training point. Supervised Machine learning is the process of parametrizing a function $f(\vec{x})$ such that:

$$r = f(\vec{x})$$

In a classic example of supervised learning [34] we are given the weather conditions ($\vec{x}$) on a few days of the month and told whether or not Joe played tennis on those days (r). Using this data, a machine learner devises a function, $f(\vec{x})$, that predicts whether Joe will play tennis on a new day, given the weather conditions. The function determined by a machine learning algorithm is called a classifier. The study presented in this dissertation relies on supervised learning, but unsupervised learning techniques could be used as preprocessing steps to this system.

Unsupervised learning is used when the categories (r) of the data are unknown, and sometimes even the number of categories is unknown. Only data points are supplied. An example of a domain that generates unsupervised training data is consumer spending records. Consumer spending data contains data points that represent customer transactions at a store and the items purchased. Unsupervised learning will identify groups of transactions that are similar. Then experts can evaluate the groups and extract the commonalities that characterize the groups. For example, perhaps we learn from a cluster of similar transactions that people who buy apple juice often also buy diapers. This new knowledge can be used to refine product placement within stores, and store owners might choose to place a small apple juice display by the diapers. The store owners suspect that this will increase the chance that a customer who has just stopped into the store to buy diapers will also decide to buy apple juice. With the advent of the store-specific points card, chains are now able to cluster not only transactions, but also people and all of their transactions over time. This can be viewed as an innovation that has made shopping easier, or a way to part you and your money. In any case, it is an example of unsupervised learning.

2.1.1 Evaluation

How can we tell if our machine learning algorithm has correctly modeled the process that generated our training set? For example, if we generate a set of rules that tell us whether Joe will play tennis on a given day, how can we tell if the rules are right? We could apply our rules to today's weather conditions, then call Joe and ask him if he is planning on swinging his racket. But then, what should we do if the prediction we make is wrong? How many wrong predictions are acceptable? At what point should we rethink our rules? If we develop two sets of rules that model Joe's tennis playing habits, how can we compare them? These questions all require a standardized and fair method for testing classifiers and measuring their performance.

Cross-Validation

Consider a training set with two class labels, 30 positive and 70 negative training instances. If we train a classifier on all 100 training instances, and then test it on the same 100 training instances we

would expect the classifier to perform very well. This is comparable to giving students an algebra test with answers ahead of time and allowing them to memorize it before taking a test comprised of the same questions, though possibly in a different order. No learning is required here, simple recall will yield a perfect score. Schools teach (and test) their pupils by covering example problems that display mathematical concepts. They test using new questions that are similar to, but not the same as the questions used to teach. To succeed, students must learn and extract concepts, instead of methodically memorizing equations and answers[2]. We would like to train and test a machine-learned classifier in a similar way.

Let us return to our 100 instance training set. If we follow the student analogy we would like to set aside a portion of the training set to use for testing, say 10% or 10 instances. We want to construct a training and testing set from the original set of instances. We would like the sets to be randomly selected, but pure randomness causes problems. If we split the groups randomly it is possible that our training set will contain only negative training instances. In that situation the trained classifier that we test will have seen 30 positive training instances, and 60 negative. When we test that classifier we will see how well it does on the negative training instances, but not on the positive instances. It is important to make the split in our training set random, but also stratified. The partitions of the training set should contain equal positive and negative proportions.

Once the training data is split into several equal and stratified partitions we withhold one partition to test the data, and use the other partitions to train a classifier. After a round of testing, the computer's memory can be cleared and the process repeated, withholding a different partition and using the other partitions to train. In "k-fold cross validation", this process repeated k times, using a different partition to test each time, where k is the number of partitions [1]. Cross-validation accurately estimates prediction statistics of a classifier, since each instance is used as a test case at some point during validation. An extreme case of k-fold cross-validation, where k is the same as the number of examples in the whole training set, is called leave-one-out.

Performance Measures

When evaluating a machine learning algorithm four measurements are often taken: true positives (tp), true negatives (tn), false positives (fp), and false negatives (fn). The first word in the name of the measurement is the validity of a classifier's call on the instance, and the second word is the class the classifier chooses for a particular training instance. For example a false positive is an instance for which the classifier made an incorrect (i.e. false) classification of an instance by predicting that a training instance belongs to a positive class when it was actually a negative training instance. These measurements are often displayed in a confusion matrix (Table 2.1) where the rows are the actual classes of the instances, and the columns are the predicted classes of the instances. A false positive will show up in the lower left hand square because it is predicted to be positive by the classifier, but

[2]Thank-you to Russell Greiner for this example.

		Predicted	
		positive	negative
Actual	positive	tp	fn
	negative	fp	tn

Table 2.1: The layout of a confusion matrix, which helps to visualize a classifier's performance. Rows are the actual class labels of the instances and columns are the predicted class labels of the instances.

		Predicted	
		positive	negative
Actual	positive	120	80
	negative	13	287

Table 2.2: An example confusion matrix. Recall is 0.60, precision is 0.90 and f-measure is 0.72

is actually negative. A good classifier will have high numbers along the diagonal of the matrix, and low numbers elsewhere.

We can combine these measurements to form precision:

$$p = \frac{tp}{tp + fp}$$

which measures the performance of a classifier's positive predictions, and recall:

$$r = \frac{tp}{tp + fn}$$

which measures how many positive instances a classifier correctly labels as positive. For example the confusion matrix in Table 2.2 represents a classifier which has high precision (0.90) but low recall (0.60). Higher precision and recall results in a more reliable classifier. However, it is difficult to increase one of the two measures without decreasing the other. A classifier that has low recall can be changed to allow more positive predictions, as shown in Table 2.3. However, allowing for more positive predictions will almost certainly hurt precision, since it is likely that some of the new positive predictions our classifier makes will turn out to be false positives. Thus, F-measure, the harmonic mean of precision and recall, is often used when measuring the performance of a classifier. F-measure is defined as:

$$f = \frac{2 * p * r}{p + r}$$

		Predicted	
		positive	negative
Actual	positive	180	20
	negative	120	180

Table 2.3: A second confusion matrix using the same training set as Table 2.2 but for a classifier that allows more positive predictions. Recall is 0.90, precision is 0.60 thus f-measure is still 0.72

F-measure is one way of balancing precision and recall to achieve the most powerful classifier, when precision and recall are deemed to be of equal importance. The confusion matrices in Table 2.2 and Table 2.3 both have the same f-measure (0.72). During the folds of cross validation we record tp, tn, fp, and fn. We can then combine the results from all folds to calculate overall precision, recall and F-measure.

2.1.2 Machine Learning Algorithms

The majority of machine learning for automatic annotation is done through supervised learning, because it can take advantage of the huge amount of labeled data created by annotators. The class label predicted by a machine learning algorithm can then be used as the protein's annotation. Though there are many machine learning algorithms, I will cover only a few of the most prevalent in automatic annotation of protein sequences.

Linear Discriminants and Support Vector Machines

Consider again the classic supervised machine learning problem where the goal is to predict if Joe will play tennis on a given day based on the weather. Assume that we consider only two features, air temperature and wind speed, when making this prediction. We could plot each of our data points in 2-dimensional space, where the x-axis is temperature, the y-axis is wind speed, circles represent days Joe did play tennis and squares represent days Joe did not (see Figure 2.1). If we can draw a straight line that separates the two classes of data points into two areas of the graph the example data is *linearly separable*. Figure 2.1 is an example of data that is linearly separable.

As with all linear functions (of 2 variables) we can represent the function in the form:

$$f(x, y) = w_1 x + w_2 y + b$$

and the line that satisfies the equation:

$$f(x, y) = 0$$

Then, given a new day we can enter the temperature (as x) and wind speed (as y) into the function. If $f(x, y) < 0$, we predict Joe will play tennis as the point lies above the line and conversely, if $f(x, y) > 0$, we predict Joe will not play tennis because the point lies below the line. The line that we draw is called a linear discriminant.

When data is not linearly separable it is not possible to draw a line that perfectly divides the space, but we may wish to find the equation of a line that best separates the data points. A Support Vector Machine [45, 1] (SVM) with a linear kernel is a special kind of linear discriminant that draws a "soft" line [12]. A soft line is one that separates those data points in the training set that are linearly separable with a line that is maximally marginal, and minimizes the number of points that are not linearly separable. A maximally marginal separator divides the space so that the distance between the separator and the closest separable points in each class is maximized. This allows for a linear

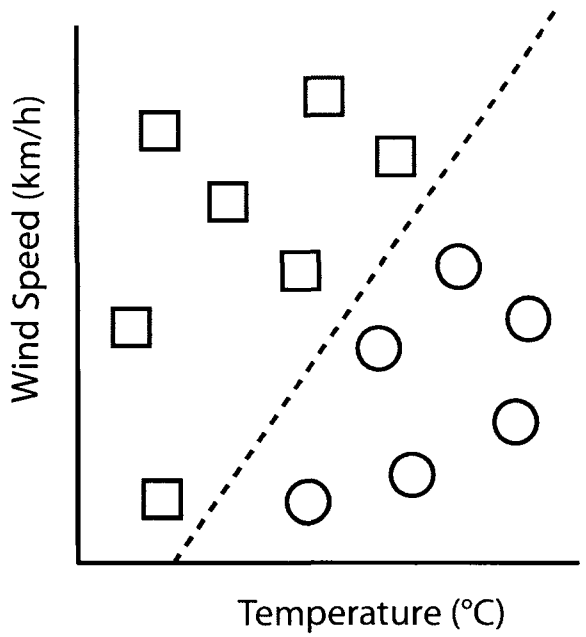

Figure 2.1: The classic supervised learning example of playing tennis, graphed in two dimensional space, the x-axis is temperature and the y-axis is wind speed. Circles indicate times Joe did play tennis and squares represent the times Joe did not play tennis. The dashed line represents the equation $f(x, y) = 0$ where x is temperature and y is wind speed. The function $f(x, y) = 0$ can be used to predict whether Joe will play tennis on a given day.

---

discriminant to be drawn even if the training data is noisy. As will be discussed in Section 3.1.1, SVMs are particularly suited to training data created from text, thus I use SVMs in this study.

An SVM learns a weight corresponding to each of the features for the data points in a training set, and an offset. For instance, in the playing tennis example the learned parameters are incorporated into a formula that describes the separation of the classes:

$$\begin{aligned} \vec{w} \cdot \vec{x} + w_0 &\geq +1 \quad \text{for play\_tennis} \\ \vec{w} \cdot \vec{x} + w_0 &< +1 \quad \text{for not\_play\_tennis} \end{aligned}$$

where $\vec{w}$ is the vector of weights that is the same dimension as the feature vector $\vec{x}$, and w_0 is the offset of the line. This line (or plane or hyperplane in higher dimensions) is "optimally separating" meaning that the distance between the separable training instances and the hyperplane is made as large as possible, which typically reduces error on unseen training instances.

Although I have discussed only linear discriminants in this section, SVMs can be trained with different kernels, meaning that they are altered to find separating hyperplanes of higher degrees. Most SVM packages have support for polynomial kernels of variable degrees, radial basis functions and sigmoid functions. These functions perform better on data that is not linearly separable, meaning that a flat hyperplane cannot separate the classes. However, the flexibility of these kernels means that they are prone to over-fitting data, as they move and adjust to fit what would be considered

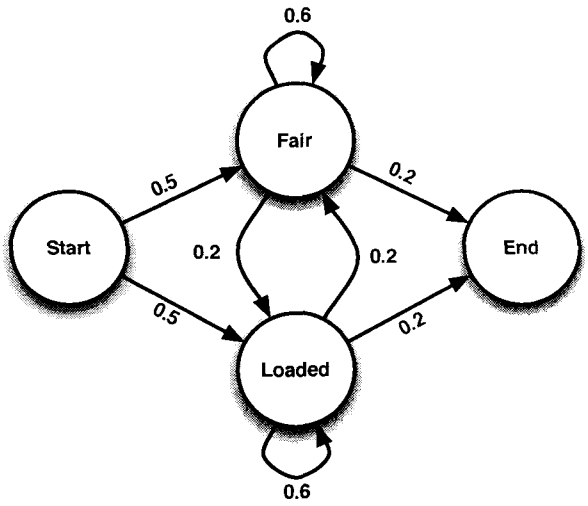

Figure 2.2: A diagram representing the transition of dice states at the "Occasionally Dishonest Casino". There are two dice states: Fair and Loaded, and two dummy states: Start and End which represent the beginning and end of a dice game. Each round of dice rolling starts in the Start state, and ends in the End state. Dice can be either Fair or Loaded, and can be switched at any time during a round. The probability of moving between the states is shown with labeled arrows.

---

outliers by less flexible functions.

Hidden Markov Models

The basis of a hidden Markov model is a Markov process [1]. A Markov processes exists in one of a set of n distinct states $S_1, S_2, \ldots S_n$ at any time, and it travels through the states as time passes[3]. Let the current state of the process at time $t = 1, 2, \ldots$ be defined as q_t. Then we have $q_t = S_i$ for some i. A Markov process moves to a new state at each time point with a probability dependent entirely on the process' history of states:

$$P(q_{t+1} = S_j) = P(q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_h, \ldots)$$

A first order Markov process is one where the probability of moving to a state at time t depends only on the state of the process at q_{t-1}, i.e. the process has a memory of one state. This can be represented as:

$$P(q_{t+1} = S_j) = P(q_{t+1} = S_j | q_t = S_i)$$

This means that the state at time q_{t+1} is independent of all states at times $q_{t-i}, i \geq 1$. In a second order Markov process the state at time q_{t+1} depends on the last two states q_t and q_{t-1}. Generally, an nth order Markov process conditions on the states at times $q_t, q_{t-1}, \ldots, q_{t-n+1}$.

[3]Time is used in this example, but any dimension can take its place. As we will see, a Markov process can also model the amino acids in a protein, in which case the process will pass through the states as we progress along the protein strand. In the protein example, time is replaced by the location on the string of amino acids $1, 2, \ldots$

	Probability	
Number Rolled	**Fair**	**Loaded**
1	0.167	0.90
2	0.167	0.02
3	0.167	0.02
4	0.167	0.02
5	0.167	0.02
6	0.167	0.02

Table 2.4: Probability of a particular number being rolled at the "Occasionally Dishonest Casino", given the state of the dice. All numbers are equally likely if the fair dice are being used, but rolling a one is most likely if the loaded dice are used.

A hidden Markov model (HMM) is a Markov process where the current state is unknown, but in each state the process emits observable signals with certain probabilities. An example of a hidden Markov model is the "Occasionally Dishonest Casino" [14]. At the Occasionally Dishonest Casino people can play games with dice, and the dice have two states: fair and loaded (see Figure 2.2). The casino can choose to switch the dice at any moment. The probability of the Casino switching the dice is called the transition probability. The probability that a certain number $\{1..6\}$ is rolled at any time is called the emission probability. Example emission probabilities are given in Table 2.4. If the casino is using the fair dice, each of the six numbers on the dice appear with equal probability ($\frac{1}{6}$). But, when the casino is using the loaded dice, certain numbers show up more often than others, meaning that the emission probabilities of some numbers are higher than those of other numbers. Since the two sets of dice appear identical we cannot tell which state ($loaded$, $fair$) the dice are in, thus the state is hidden. We can use the frequency of the numbers rolled (the emissions of the states) to predict the hidden state of the dice. For example, let us say we are at the Occasionally Dishonest Casino and we roll the following series of numbers:

$$1, 2, 3, 4, 5, 6, 1, 1, 1, 1, 1, 1, 1$$

Using the transition probabilities in Figure 2.2 and the emission probabilities in Table 2.4 we can use a dynamic programming algorithm called the Viterbi algorithm [46] to calculate the most probable sequence of hidden dice states, given the observed numbers. The most probable state sequence is:

$$L, F, F, F, F, F, L, L, L, L, L, L, L$$

where L represents a number produced by a loaded die, and F represents a fair die roll. The probability of this path is 0.255, calculated by multiplying the transition probabilities and emission probabilities for the number sequence[4].

But what if we are visitors at the Occasionally Dishonest Casino, and do not know the emission and transition probabilities? If we observe many thousands of rolls of the dice, we can use the Baum-Welch algorithm [7] to estimate these probabilities. Thus, if we watched the dice long enough at the

[4]Thank you to Dekang Lin for his Viterbi implementation, used here.

Occasionally Dishonest Casino, we could learn the model that governs the switching of the loaded and fair dice. Using that model we could then predict when the dice have been changed by watching the numbers that are rolled[5].

2.2 Machine Learning in Bioinformatics

2.2.1 Sequence-Based Prediction

Sequence-based methods of prediction use only the amino acid sequence of a protein when making predictions. All sequence-based methods build models of the types of amino acids and patterns of amino acids that are found in proteins with particular properties, but their methods for creating and representing these models differ. Because these algorithms use no outside information, they are the simplest methods for predicting protein properties.

Sequence Similarity

One of the simplest (though not the most accurate [31]) ways to annotate a new protein is to find the annotated protein with the most similar amino acid sequence and transfer its annotation to the new protein. But how do we determine if two proteins are similar? Say we have two proteins of lengths n_1 and n_2; we could lay them side by side and compare them, amino acid by amino acid, and assign a "percent identity" score based on the total number of matches. This would take $O(n_1)$ time where n_1 is the length of the shorter sequence. But what if the proteins have higher percent identity if we begin the protein comparison with the first amino acid of the shorter protein and the second amino acid of the longer protein? A brute force approach to cover all possible full alignments will make $O(n_1)O(n_2 - n_1)$ comparisons. In nature, occasionally long sequences of DNA are deleted, resulting in deletions of entire stretches of amino acids in proteins, but no loss of function. To model this we have to allow for gaps in our alignment, where we skip over a few amino acids in one protein and pick up our amino by amino comparison after the gap. Once a gap is introduced, the problem really is one of optimally aligning two smaller sequences (created by splitting one protein with a gap) simultaneously with a longer sequence. Approaching the task, called pair-wise sequence alignment, in this way transforms it into a dynamic programming problem, solvable in $O(n_1 n_2)$ time.

It is important to note that the effects of amino acid substitutions are less severe to the function of a protein if the substituted amino acid has similar properties. For example, isoleucine is often substituted by valine in proteins without a change in the protein's function. This substitution without phenotypic change is possible because isoleucine and valine share many of the same physical properties (hydrophobic and non-polar) and thus cause little change to the proteins overall conformation.

[5] Unfortunately the Occasionally Dishonest Casino has a strict "no computers" policy.

Because amino acids naturally substitute each other with differing frequencies, scoring matrices built from the observed frequencies of substitutions are often used to score alignments. A scoring matrix has a row and a column for each amino acid. Each cell of the matrix represents a score for aligning the row amino acid with the column amino acid in a sequence alignment, based on the similarity of those amino acids. The matrices have the highest scores for identical amino acids, and decreasing scores as the properties of pairs of amino acids diverge. A commonly used matrix is the BLOSUM (Blocks Substitution Matrix).

A BLOSUM matrix is derived by clustering sequences that have a certain level of identity and observing the amino substitution rate in each cluster. For example, the BLOSUM62 matrix is created using the amino acid substitution frequency of protein cluster with 62% identity.

A number representing the frequency of substitution of the row and column amino acids fill the cells of the matrix. For example, in the BLOSUM62 matrix isoleucine and valine have a substitution score of 3, almost as high as isoleucine aligned with itself (score of 4). In comparison the BLOSUM62 matrix assigns isoleucine and glycine a substitution score of -4 because the polar and hydrophilic molecule of glycine is unlike the non-polar hydrophilic isoleucine.

While dynamic programming allows us to more efficiently perform pair-wise sequence alignment, trying to find the most similar protein in all of Swiss-Prot using pair-wise sequence alignment takes an unreasonable amount of time. BLAST (Basic Local Alignment Search Tool) was born out of the need for faster sequence alignment, at the expense of sensitivity [2]. BLAST begins its alignment by finding "seeds" or short sequences of amino acids that appear in both the database, and the query protein. This is done quickly using a look up table of short sequences that is created beforehand. These short seeds are then lengthened by checking if the amino acids on either side of the seed match. If a sequence matches many of the seeds in a sequence from the database a full gaped alignment is performed. Experiments have shown that a full alignment of proteins using a dynamic programming method takes 36 times longer than a heuristic BLAST search [3].

Blast uses e-value as a scoring metric for the alignment of sequences. Intuitively, an e-value represents the probability that a protein would generate the given alignment to a protein in the database by random chance. More formally, e-value is defined as:

$$E = mn2^{-S'}$$

where n is the size (in amino acids) of the database, m is the length of the query sequence and S', the bit score, is defined as:

$$S' = \frac{\lambda S - \ln K}{\ln 2}$$

where S is the score of the alignment, calculated using a scoring matrix, and λ and K are statistical parameters tuned by the creators of BLAST.

Sequence alignment can also be performed on many sequences at the same time to form a multiple sequence alignment. In a multiple alignment, sequences are pairwise aligned, often using an

```
CLUSTAL W (1.83) multiple sequence alignment


Trout             ----------VEWTDAEKSTISAVWGKVN--IDEIGPLALARVLIVYPWTQRYFGSFGNV 48
Rockcod           ----------VEWTDKERSIISDIFSHMD--YDDIGPKALSRCLIVYPWTQRHFSGFGNL 48
Human             ---------MVHLTPEEKSAVTALWGKVN--VDEVGGEALGRLLVVYPWTQRFFESFGDL 49
Human-sickle      ----------VHLTPVEKSAVTALWGKVN--VDEVGGEALGRLLVVYPWTQRFFESFGDL 48
Gorilla           ---------MVHLTPEEKSAVTALWGKVN--VDEVGGEALGRLLVVYPWTQRFFESFGDL 49
Spider-Monkey     ----------VHLTGEEKAAVTALWGKVN--VDEVGGEALGRLLVVYPWTQRFFESFGDL 48
Horse             ----------VQLSGEEKAAVLALWDKVN--EEEVGGEALGRLLVVYPWTQRFFDSFGDL 48
Pig               ----------VHLSAEEKEAVLGLWGKVN--VDEVGGEALGRLLVVYPWTQRFFESFGDL 48
Cow               ----------MLTAEEKAAVTAFWGKVK--VDEVGGEALGRLLVVYPWTQRFFESFGDL 47
Deer              ----------MLTAEEKAAVTGFWGKVD--VDVVGAQALGRLLVVYPWTQRFFQHFGNL 47
Gull              ----------VHWSAEEKQLITGLWGKVN--VADCGAEALARLLIVYPWTQRFFASFGNL 48
Lamprey           --PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQEFFPKFKGL 58
Sea-Cucumber      XGGTLAIQAQGDLTLAQKKIVRKTWHQLMRNKTSFVTDVFIRIFAYDPSAQNKFPQMAGM 60
                              :  ::  :   :  :          : : :   * :*. *  : .:

Trout             STPAAIMGNPKVAAHGKVVCGALDKAVKNMGN---ILATYKSLSETHANKLFVDPDNFRV 105
Rockcod           YNAEAIIGNANVAAHGIKVLHGLDRGVKNMDN---IAATYADLSTLHSEKLHVDPDNFKL 105
Human             STPDAVMGNPKVKAHGKKVLGAFSDGLAHLDN---LKGTFATLSELHCDKLHVDPENFRL 106
Human-sickle      STPDAVMGNPKVKAHGKKVLGAFSDGLAHLDN---LKGTFATLSELHCDKLHVDPENFRL 105
Gorilla           STPDAVMGNPKVKAHGKKVLGAFSDGLAHLDN---LKGTFATLSELHCDKLHVDPENFKL 106
Spider-Monkey     STPDAVMSNPKVKAHGKKVLGAFSDGLAHLDN---LKGTFAQLSELHCDKLHVDPENFRL 105
Horse             SNPGAVMGNPKVKAHGKKVLHSFGEGVHHLDN---LKGTFAALSELHCDKLHVDPENFRL 105
Pig               SNADAVMGNPKVKAHGKKVLQSFSDGLKHLDN---LKGTFAKLSELHCDQLHVDPENFRL 105
Cow               STADAVMNNPKVKAHGKKVLDSFSNGMKHLDD---LKGTFAALSELHCDKLHVDPENFKL 104
Deer              SSAGAVMNNPKVKAHGKRVLDAFTQGLKHLDD---LKGAFAQLSGLHCNKLHVNPQNFRL 104
Gull              SSPTAINGNPMVRAHGKKVLTSFGEAVKNLDN---IKNTFAQLSELHCDKLHVDPENFRL 105
Lamprey           TTADELKKSADVRWHAERIINAVDDAVASMDDTEKMSMKLRNLSGKHAKSFQVDPEYFKV 118
Sea-Cucumber      S-ASQLRSSRQMQAHAIRVSSIMSEYVEELDS--DILPELLATLARTHDLNKVGADHYNL 117
                    .  :  .  :  *.  :   .   :  :..   :            .   *..: :.:

Trout             LADVLTIVIAAKFGASFTPEIQATWQKFMKVVVAAMGSRYF 146
Rockcod           LSDCITIVLAAKMGHAFTAETQGAFQKFLAVVVSALGKQYH 146
Human             LGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH 147
Human-sickle      LGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH 146
Gorilla           LGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH 147
Spider-Monkey     LGNVLVCVLAHHFGKEFTPQLQAAYQKVVAGVANALAHKYH 146
Horse             LGNVLVVVLARHFGKDFTPELQASYQKVVAGVANALAHKYH 146
Pig               LGNVIVVVLARRLGHDFNPDVQAAFQKVVAGVANALAHKYH 146
Cow               LGNVLVVVLARNFGKEFTPVLQADFQKVVAGVANALAHRYH 145
Deer              LGNVLALVVARNFGGQFTPNVQALFQKVVAGVANALAHKYH 145
Gull              LGDILIIVLAAHFAKDFTPDSQAAWQKLVRVVAHALARKYH 146
Lamprey           LAAVIADTVAAG----------DAGFEKLMSMICILLRSAY- 149
Sea-Cucumber      FAKVLMEALQAELGSDFNEKTRDAWAKAFSVVQAVLLVKHG 158
                  :.  :  .:              : * .  :   :   :
```

Figure 2.3: A multiple sequence alignment of the beta chain hemoglobin molecule of several organisms (example courtesy of Dr. Rick Hershberger, The Bioactive Site) as performed by clustalW [20]. The last row of each block of the alignment is annotated with "*" for complete consensus, ":" for highly conserved amino acids, and "." for semi-conserved positions. A "-" within a protein sequence signifies a gap in the alignment.

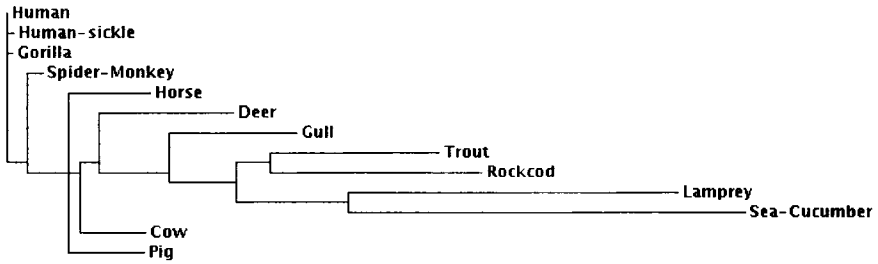

Figure 2.4: A phylogenetic tree representing the alignment in Figure 2.3.

---

approximate alignment for speed. A multiple alignment is built by progressively aligning the proteins, beginning with those deemed most similar by the pairwise alignment. The final result is an alignment of all the proteins. ClustalW [20] is one program that can be used to perform multiple alignments. Figure 2.3 shows the output of a ClustalW alignment of hemoglobin beta chain proteins from several organisms (example courtesy of Dr. Rick Hershberger, The Bioactive Site). The last row of each block of the alignment is annotated with "*" for complete consensus, ":" for highly conserved amino acids, and "." for semi-conserved positions. A "-" within the protein sequence signifies a gap in the alignment.

Alignments of multiple proteins that perform the same function will elucidate the regions that are conserved by selective pressure, and likely encode the function of a protein. Because they encode selective pressure, and because time brings increased mutations, the results of multiple sequence alignments can be used to make phylogenetic trees. A phylogenetic tree is a tree that encodes the evolution of species. The path in a phylogenetic tree will be shorter between species that are closer evolutionarily than those that are less similar. A phylogenetic tree constructed from a multiple sequence alignment assumes that those organisms with the most similar copies of a protein are also the closest evolutionarily, and thus should be closest on the tree. An example of such a tree, as created by ClustalW using beta hemoglobin proteins, is shown in Figure 2.4.

PROSITE

Sites of proteins that perform similar functions, such as catalyzing similar reactions or binding similar molecules, often share stretches of amino acids that are identical or very similar. PROSITE [38] is a collection of motifs that fall into two categories: patterns and profiles. Patterns are usually 10-20 amino acids in length and are well conserved in order to maintain the function that stretch of the protein serves. To create a pattern, segments of proteins known to perform a particular function are aligned and a pattern is constructed from the aligned amino acids. For example, the cytosol

aminopeptidase signature (PS00631) is encoded as

$$[NS] - T - D - A - E - G - R - [LV]$$

which uses the one-letter amino acid abbreviations to encode the amino acids of the pattern. A protein matching the example pattern can begin with an asparagine (N) or a serine (S) followed by threonine (T) - aspartic acid (D) - alanine (A) - glutamic acid (E) - glycine (G) - arginine (R) and ending with a leucine (L) or a valine (V). This pattern was created by aligning all known cytosol aminopeptidases, enzymes that break down old proteins in the cell so that new proteins can be made from their amino acids. This octapeptide is perfectly conserved across all of the known cytosol aminopeptidases.

PROSITE profiles [38] were developed to address the major flaw of PROSITE patterns: that patterns do not allow for mismatches. Profiles allow for mismatches in a pattern alignment. A profile is similar to a pattern in that it represents a sequence of amino acids, but each amino acid match position is followed by an optional insert position. Each match position has a deletion penalty and an associated weight for every possible amino acid associated with that position. These weights and penalties allow a score to be assigned to the sub-sequence of a protein that matches a profile. High-scoring matches to a profile indicate a good match to the underlying pattern, and can identify proteins that a pattern alone would have missed.

Pfam

Pfam [6] is a collection of protein families modeled by multiple sequence alignment and HMMs. Pfam HMMs predict whether or not a stretch of amino acid sequence does or does not serve the function of, or have a property belonging to, a specific protein family. The statistical nature of the HMM gives them more flexibility than the regular expressions seen in PROSITE. Pfam creates two HMM models for each family. The first is a seed alignment built from a set of proteins that an expert identifies as belonging to a family. This seed alignment is used to search Swiss-Prot and Trembl for other similar proteins. The group of proteins found using the seed HMM are realigned and checked by an expert. If the final alignment passes expert scrutiny it is used to create a final, more general HMM. Version 18 of Pfam has 7973 families, organized into clans that represent the evolutionary origin of the protein families [16].

TMHMM

TMHMM [40] is an HMM model that predicts transmembrane helices in amino acid sequences. Transmembrane helices are parts of proteins that pass through the phospholipid bilayers of cells. TMHMM is used to predict which part of the protein passes through the bilayer, and also the topology of the protein, i.e. which parts of the protein are on the external side of the membrane, and which parts are internal. The hidden states of the HMM and the possible transitions between the states were chosen in a way that reflects the biological system that governs transmembrane helices.

An example of these biological restrictions is that transmembrane regions must have between 5 and 25 amino acids. TMHMM is so widely accepted as an accurate transmembrane predictor that it was used to automatically annotate the transmembrane regions of every entry in the Swiss-Prot database.

MultiLoc

MultiLoc is a system that combines the output of several SVMs trained on the protein sequences themselves to make predictions about the subcellular localization of proteins [23]. There are 4 sub-classifiers trained to predict specific properties of the protein:

- SVMTarget makes a localization prediction based on the amino acids at the N-terminal of the protein sequence. These amino acids are often used to target the protein to a particular area of the cell and thus are good predictors of localization.
- SVMaac uses the amino acid composition of the whole protein to predict localization
- SVMSA was devised to detect signal anchors (SA) that often indicate that a particular protein is part of a membrane.
- Motif Search searches for motifs in the protein sequence. The motif patterns come from the PROSITE and NLSdb databases.

MultiLoc uses a super-classifier, and SVM, which combines the output of all of the sub-classifiers to make one final prediction. The overall accuracy of the system is about 75%, a big improvement over its predecessor, PSORT, which yielded accuracy around 59%.

2.2.2 Homology-Based Prediction

A different approach to protein annotation is to use protein similarity, or homology, to find annotated proteins that are similar to an unannotated protein. The amino acid sequence of a protein is the basis of its function, thus proteins with very similar sequences are likely to have the same function. BLAST is often used to find proteins that share a high level of sequence similarity. This approach works well for orthologues: proteins with sequences that are identical or very close to identical, but appear in different organisms. However, occasionally two proteins that have very similar amino acid sequences have diverged enough that the two serve different functions in the cell. Thus, it is not always enough to label a new protein with the annotations of the most similar protein in Swiss-Prot, and more intelligent methods have been devised to deal with this issue.

Proteome Analyst

Proteome Analyst (PA) [43] is a web-based tool which uses BLAST and Swiss-Prot together to predict a protein's subcellular localization and molecular function. PA uses BLAST to compare a protein to the Swiss-Prot database. PA identifies the top 3 BLAST hits (with a score above a

predefined threshold) and extracts tokens from their annotations. PA uses these annotations and SVM technology to choose the best class label for the query protein. The feature vector $\vec{x}$ contains the keywords extracted from the 3 closest BLAST hits, and the class label is whether or not a protein has a particular function or a particular localization. This is not as simple as transferring the annotations of the most similar protein, a process called *1-nearest-neighbour*. 1-nearest-neighbour has been shown to produce results poorer than PA's SVM mixture of annotations when applied to the subcellular localization classification task [31]. The machine learning SVM step can be viewed as a mixing of annotations to create an output annotation, and the nature of that mixing is based on real world examples. PA has been used to automatically annotate all known proteins in 47 organisms, totaling over 330,000 sequences.

2.2.3 Combinations of Homology and Sequence

Both homology- and sequence-based prediction methods have their flaws. Homology-based models work only when there is at least one good annotated match to a new protein. Sequence-based methods do not always return a correct prediction, even if the protein is very similar to other proteins in a family. Some methods use both homology and sequence-based predictions so that the shortcomings of one approach are compensated by the other.

PSORTb

PSORTb (version 2.0) is a web-based application that predicts subcellular localization for bacterial proteins. PSORTb works by mixing the predictions of several sub-classifiers into one final prediction. The first sub-classifier works using BLAST and a custom database of proteins with known subcellular localization. This sub-classifier selects proteins returned by BLAST that have an e-value less than 10^{-10} and length within 20% of the query protein. This first sub-classifier simply returns the subcellular localization of the top hit (1-nearest-neighbour), and if the top hit is 100% identical to a protein in the database, that localization is returned and none of the other sub-classifiers are used. The second sub-classifier is an SVM. The feature vector for this SVM contains the presence or absence of frequently occurring subsequences in the protein. The classifier returns a yes/no answer for each possible localization. PSORTb also uses the output of PROSITE motif finders that have been shown to return 0% false positive rate. The remaining classifiers identify stretches of amino acids that strongly indicate one of three localizations:

- A classifier that identifies beta barrels that indicate a protein is likely part of the outer membrane.
- A classifier that recognizes alpha helices, which are strong indicators that a protein is active in the inner membrane.

- A classifier that identifies signal peptides, which strongly suggest that a protein is destined for a location outside of the cell (i.e. to be secreted by the cell).

The results of these classifiers are combined to create a score for each possible localization. This method is very precise, but misses a lot of proteins that should be labeled with certain localizations (has low recall).

2.3 Conclusion

The advent of computer-generated annotations for proteins has lessened the load on human annotators, and some bioinformatics tools are trusted enough by the biological community that their annotations have been entered into high-quality databases such as Swiss-Prot. Still, a large portion of biological databases remains unannotated, and many tools' annotations are still not accurate enough to be accepted by biologists. However, many of the proteins that remain unannotated in Swiss-Prot have been the subject of biological research. Swiss-Prot entries contain links to the abstracts of journal articles that discuss relevant research. While many computational methods use words contained in the annotations of the Swiss-Prot records, few follow links to journal abstracts and attempt to incorporate that knowledge. This represents a huge untapped resource, which could increase the coverage of annotations in databases like Swiss-Prot, and could also improve prediction methods like PA, which currently only use the information stored directly in Swiss-Prot entries. However, the language of biological journal abstracts represents a real challenge. Unlike many annotation fields, abstracts are free-form text and so they display a high degree of linguistic variation. Despite the difficulties of the task, it is imperative to effectively utilize every piece of information recorded about a protein in order to provide the best computer-generated annotations to the biological community. The following chapter discusses computational techniques that can be used to gather the knowledge present in biological journal abstracts to improve automatic annotation of proteins.

Chapter 3

Introduction: Natural Language Processing

Computer-human interfaces have taken many forms since the computer's inception - from punch cards to keyboards to video game controllers and beyond. Interacting with a computer is becoming more intuitive every day. The ultimate in human computer interaction will be the day that we communicate with our computers like we would with another human, using natural language.

Natural Language Processing (NLP) is a mixture of linguistics and computing science, especially machine learning. Research in NLP enables us to gain a deeper understanding of the way humans use language. This deeper understanding can be used to help computers understand language and to further areas of theoretical and applied linguistics. This chapter focuses on the former application, the way in which a deeper understanding of language improves a computer's ability to understand language.

3.1 Areas of text processing

NLP comes in many different flavours. Some researchers are interested in the structure of sentences and the roles of the words in sentences. Some are interested in the similarities in word usage. Some study the similarities of languages themselves. I discuss here the two sub-topics of NLP that are most readily applied to the task of using the text in journal abstracts to aid in automatic protein annotation.

3.1.1 Document classification

Document classification is a supervised learning problem that assigns a set of predefined class labels to a text document. A familiar example of this task appears in the context of the average newspaper, where text documents (newspaper articles) are divided into sections of the paper (sports, world news, financial news). Specifically, my task is to assign a subcellular localization category (e.g. nuclear) to a biological journal abstract. In this study, once a document or set of documents has been classified, the assigned class label can be used as an annotation for the associated protein.

In document classification, documents are treated as "bags of words". This means that the document is broken down into words, using a process like white-space tokenization. These words are collected together to create the feature vector $\vec{x}$ containing words as features, and the counts of these words as the feature values. This method does not encode any of the spatial relationships of the words, and so some of the semantic content could be lost. For example, a "bag of words" does not encode if two words appear next to each other in the text, or if even appear in the same sentence. Despite this, the "bag of words" method is very popular and has proven successful for the classification of biological literature [41, 22].

To improve text classification systems, some preprocessing steps are often performed on the words produced by tokenization. Often, to decrease the feature space (i.e. dimension of $\vec{x}$) and to more closely capture the meaning of words, a process called stemming is applied to text documents. Stemming removes suffixes from words using a set of language-specific rules. The Porter Stemmer [33] uses a set of rules that removes suffixes like "ed", "ing" and "s" from English words. For example, the Porter Stemmer will derive the stem *connect* from any of the words *connects, connected, connection, connecting*. Thus, if an author is using any of a set of words that have the same stem, stemming will encode them all as the same concept. Stemming is a process that does not take into account properties of the root of the word when stripping the suffix, and considers only the standard ways in which English appends suffixes to words. This simplicity allows stemmers to handle scientific words that may not appear in standard dictionaries. However, Stemmers are not perfect. Some cases, like "flew" the past tense of "fly" will not be reduced to its base form because it is an irregular verb, and is unrecognizable to the stemmer.

Once the word boundaries in a sentence have been identified and words have been reduced to their stems, word counts are collected. It is at this point that we must answer an interesting and important question in document classification: how should text be represented as a feature vector? Is the raw count of the words enough, or should they be manipulated in some way to represent the word's distribution within the collection? Leopold and Kindermann performed a very thorough survey of this area in which they tested two categories of frequency transformations: raw frequencies and logarithmic frequencies [29]. The raw frequency is the count of the times a word appeared in the text and the logarithmic frequency is $\log(1 + raw\ frequency)$. Logarithmic frequencies compress the range of numbers seen as values in the feature vector. The difference between not seeing a word in a document and seeing it once in a document is much more important than the difference between seeing a word 200 times in a document and seeing it 201 times in a document [21]. A logarithmic transformation reflects this change in importance by compressing the range of feature values for very high word counts.

Frequency representations are often combined with *importance weight measures* to represent a word's significance. For example, if a word appears five times in every document, that word is not a good indicator of the document's class label. However, if the word appears five times, but only in a

few documents, that word may be a better indicator of a document's class label. Importance weight measures help to represent the probability that a word can help a classifier distinguish between class labels. Two importance weights are explored in [29]: inverse document frequency and redundancy. Inverse document frequency (idf) is a weight that puts more importance on a word if it is found rarely in the set of all documents. Formally, idf is:

$$idf = \log \frac{N}{df_k}$$

where N is the number of documents in a collection and df_k is the number of documents in the collection that contain word w_k. Inverse document frequency was first proposed by Salton and Buckley (1998) and has been used extensively in various forms for text categorization [27, 41]. If a word is seen several times in a protein's abstract and it is seen many times in the set of all other documents, the word's weight will be smaller. Conversely if the word is seen several times in a document and rarely in any other documents, its weight will be larger. In this way, TFIDF down-weights common words, and gives more weight to less common words. The drawback of using idf is that it takes into account only the ratio of documents that contain a certain word, and not the frequency of that word in the documents. Thus Leopold and Kindermann explore an importance weight measure called redundancy which represents how much the distribution of a given word across the documents in a set deviates from a uniform distribution [29]. Formally, redundancy is

$$r_k = \log N + \sum_{i=1}^{N} \frac{f(w_k, d_i)}{f(w_k)} \log \frac{f(w_k, d_i)}{f(w_k)}$$

where $f(w_k, d_i)$ is the number of times word k appears in document i, $f(w_k)$ is the number of times word k appears in all documents and N is the number of documents. If a word appears the same number of times in all documents the redundancy weight for that word will have a value of zero, thus cancelling out the frequency of the word when combined in the feature vector. If a word appears in only one document its redundancy weight will evaluate to $\log N$. In general, the more uniform the word's distribution among the documents, the lower the redundancy weight. The more skewed the word's distribution, meaning it appears much more frequently in some documents than others, the higher the redundancy weight. Redundancy is related to entropy, which is used in information theory.

Different types of normalization are explored in Leopold and Kindermann's study. They use L_1 and L_2 normalization, as well as no normalization at all (raw TFIDF and redundancy values). L_1 normalization involves dividing each feature value by the sum of all feature values. L_2 normalization involves dividing each feature value by the sum of all feature values, which creates a unit length feature vector.

Leopold and Kindermann go on to explore 30 combinations of these text representations. They are made by combining each of:

- raw frequency, with L_1 and L_2 normalization

- log frequency, with L_1 and L_2 normalization
- raw frequency with redundancy along with with L_1 and L_2 normalization
- log frequency with redundancy along with with L_1 and L_2 normalization
- raw frequency with idf along with L_1 and L_2 normalization

with one of three SVM kernels (linear, 2nd order polynomial and Gaussian radial basis function). They tested each of these combinations on two newspaper article corpora, one English (Reuters-21578) and one German (Frankfurter Rundschau). They noted that while idf and L_2 normalization is often used for document classification, it actually performs worse than the combination of redundancy and L_2 normalization. They also observed that L_1 normalization tended to work better for longer text collections. As a general observation, it did not appear that the SVM kernel had a strong effect on the performance of the resulting text classifiers. It should be noted, however, that the idiosyncrasies of the language style used in a specific corpus will affect the suitability of a particular combination. Often, several combinations should be tested; there is no one optimal combination for all corpora.

There is another important question to consider when embarking on a document classification task. What machine learning algorithm will perform the best with respect to precision, recall and f-measure? Textual data often creates training input with many features (i.e. the dimension of $\vec{x}$ is large), and the features are sparse (i.e. many of the entries of $\vec{x}$ are zero and there are many relevant features) [27]. For example, the data created for this dissertation using the PA data set has close to 60,000 unique features (words) and each training instance has, on average, 156 positive features. That means that most training instances will have feature vectors made up of 99.7% zeros. This data profile does not suit every kind of machine learning algorithm, as the running times of many algorithms are dependent on the dimensionality of the feature vector. Dumais et al. [13] explored the application of several machine learning algorithms to document classification, including SVMs, naïve Bayes and Bayes nets[1]. SVMs were found to be the most computationally efficient and to have the highest precision/recall break-even point (BEP, the point where precision equals recall). Thus, since the algorithm is most suited to the data profile and tends to perform very well, SVMs are used extensively in text categorization.

Document classifiers accurately produce subcellular localization annotations, sometimes yielding recall as high as 89% and precision as high as 91% for classes of the MultiLoc data set (mentioned in Section 2.2.1 [22]). These classifiers have even better performance when supplied with additional information about the actual protein sequence. These same experiments showed improvements of as much as 32% for recall and 25% for precision when sequence information, such as amino acid motifs, was incorporated. Stapley et al. showed that incorporating the amino acid

[1] For an explanation of these machine learning algorithms, see [1]

composition of a protein (the percent amount of each amino acid in the protein sequence) raised the F-measure of a text based classifier by as much as 5% [41].

3.1.2 Named Entity Recognition

When people read text, they have no problem identifying the words in a sentence that refer to entities: people, places or things. Even when a sentence refers to a person with an unfamiliar name we can still identify that it as a name. We do this by using clues present in the structure of the sentence and the words around the name. Named Entity Recognition (NER) tries to perform this task automatically[2]. In the past, researchers have devised lists of rules that can be applied to sentences to extract named entities. Unfortunately these lists are often difficult to maintain and are training-data specific, so they often do not perform well on new data. New approaches in NER involve building statistical models allowing computers to automatically pick out entities in text [8, 32]. Newly identified entities can be used in document classification as additional features. Because named entities often span word boundaries, an NER system creates new features for document classifiers that would have otherwise been lost due to white space tokenization.

NER is particularly useful for biomedical term identification, including the identification of gene names. Lists of gene names, and gene name synonyms have been compiled, but there is a gap between the time a gene is named and the time it is incorporated into gene name lists. Thus some computer scientists employ HMMs to model the structure of the words that appear in a sentence before a named entity. In Bikel et al.'s [8] work using HMMs for NER the hidden classes consisted of seven types of entities (person, organization, etc.) and one non-entity state. There are 14 features for each word that encode things like whether the word has numeric characters, is all capital letters, or begins with a capital letter. Thus the problem becomes one of finding the most probable sequence of entity states given the words of a sentence and their characteristics.

A biomedical entity recognition workshop was organized at the Conference on Computational Linguistics (COLING) 2004 [28]. The workshop organizers used GENIA 3.02 which is a corpus of 2,000 biological journal abstracts annotated for named entity recognition [11]. The submitted programs' precision and recall were tested on 440 new, unseen abstracts. The baseline for the experiment was a simple method that simply searched in the 440 new abstracts for any named entity that also existed in the training set, comparable to looking for a protein name in a list of known protein names. The baseline method yielded 52.6% F-measure for correctly identifying the left boundary of entities, 47.7% for the right boundary, and 43.6% F-measure for correctly determining the exact boundaries (both the left and right boundaries) of a named entity. The best NER method for this task was submitted by Zhou and Su [48] and it yeilded 76.0%, 72.6% and 69.4% for left boundary, right boundary and exact boundary F-measure respectively.

Zhou and Su's algorithm for detecting named entities uses HMMs and SVMs. When HMMs

[2] An NER challenge was held at the Conference on Computational Natural Language Learning held in 2003. For an overview of the task, and a variety of approaches see [35]

are employed in practical applications, often there is not enough data to properly estimate all the probabilities of a given event, considering the states before it. Zhou and Su use an SVM to fill in this missing information. The distance of a training instance to the hyperplane is converted to a probability measure using a Sigmoid model. An SVM is trained for every entity class plus the non-entity class, and the converted probabilities of all SVM predictions are normalized. The normalized probabilities are used as input to the HMM when data to estimate the probability is unavailable.

3.2 Conclusion

The techniques discussed in this chapter represent a sampling of the most recent and effective applications of NLP to the problem of automatic annotation. While the field has made advances in the last decade, there is still room for improvement. Scientific language is complex and technical. It contains many synonyms and can be difficult for people to understand, let alone computers. In these situations it is best to use computers for what they are best at - finding patterns and processing large amounts of data. If we combine the pattern finding and data processing capabilities of computers with the extensive organization efforts biologists have invested in their research collections, perhaps we can build stronger automatic annotation systems.

Chapter 4

Methods

Biologists have gone to great lengths to organize and categorize not only their research results, but also the terminology they use to describe their research. They have amassed a large repository of proteins and associated text describing research on these proteins. In addition, biologists created the Gene Ontology (GO) that describes scientific language in a sophisticated directed acyclic graph (DAG) structure. This chapter outlines experiments whose aim is to test whether the structure of the GO can be utilized to extract more information from the huge text resources biologists have created. The workflow used to test this hypothesis is outlined in Figure 4.1. *Process a* in Figure 4.1 is described in Section 4.2 and Section 4.3 discusses *Process b*. Two of the data sets shown in Figure 4.1 (Data Sets 2 and 3) use the GO hierarchy to improve text classification.

4.1 Gathering Data

The first step in evaluating the usefulness of GO as a knowledge source for enhancing automatic subcellular localization annotation is to create a data set. This process begins with collecting a group of proteins for which subcellular localization is known (represented by the "Set of Proteins" in Figure 4.1). The Proteome Analyst (PA) group has created such data sets and, as mentioned in Chapter 2.1, used them to create very accurate subcellular classifiers using the keyword fields of Swiss-Prot entries for homologous proteins.

Proteome Analyst creates its data sets by searching for any of a set of phrases in the subcellular localization annotations of proteins in the Swiss-Prot database. They have built a mapping of commonly occurring phrases to localizations. For example the phrase "nucleolar" maps to the localization "nucleus". All the phrases that map to the "nucleus" class label for animals are given in Table 4.1. Some proteins have more than one localization. For example MOD5 [9] is a gene that encodes 3 isozymes that localize to the mitochondrion, cytoplasm and nucleus. PA's data sets allow a protein to have more than one class label. In some cases it may appear that a protein should have more than one class label, when in fact it has only one. For example Swiss-Prot version 48 has the following subcellular localization annotation for protein with accession number O03376: "subcel-

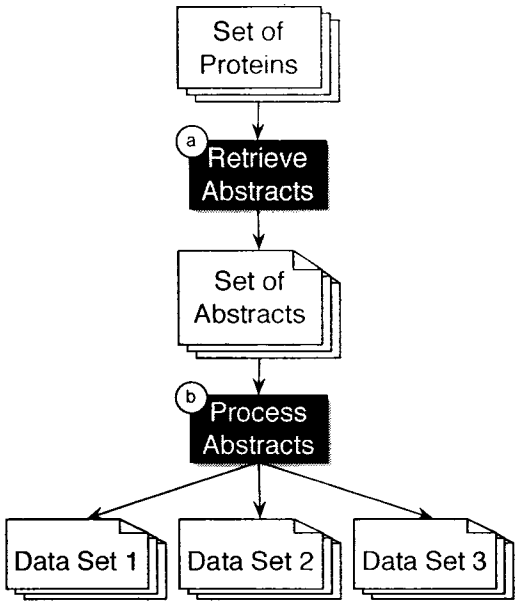

Figure 4.1: The workflow used to create data sets used in this paper. Abstracts are gathered for proteins with known localizations (process *a*). Three treatments are applied to abstracts to create three Data Sets (process *b*). The f-measures of SVM classifiers trained using these data sets are recorded and compared.

lular location: mitochondrial, possibly in the inner surface of the inner mitochondrial membrane". This annotation will match membrane phrases and mitochondrial phrases. For this reason a special class, MP, is used to denote an annotation that indicates the protein is involved in a membrane structure. If the protein has only the MP mapping, then it is called a plasma membrane protein. However, if the protein has an MP mapping and also a mapping to another localization class that involves a membrane, it is not mapped to the plasma membrane. The example above will not be mapped to the plasma membrane because the mitochondrion has a membrane. When incorporated into the training data, protein O03376 will be given only the mitochondrion class label. Thus, proteins can only be mapped to the plasma membrane class if they are not mapped to any other localization that incorporates a membrane.

In this study, PA's method of selecting proteins is used on Swiss-Prot (version 48.3) with one further constraint: the subcellular localization annotation may not be longer than four words. This constraint is introduced to avoid including proteins where the localization category was incorrectly extracted from a long sentence describing several aspects of localization. For example, consider the subcellular annotation "attached to the plasma membrane by a lipid anchor". In an animal cell, this could mean that the protein's functional components are either cytoplasmic or extracellular, depending on which side of the plasma membrane the protein is anchored. PA's simple parsing scheme could mistake this description as meaning that the protein performs its function in the plasma membrane. Enforcing a length constraint reduces the chances of including mislabeled training instances

Localization Phrase	Localization Category
nucleolar	nucleus
nucleoli	nucleus
nuclear matrix associated	nucleus
nucleus	nucleus
nuclear	nucleus
nuclear membrane	nucleus
nuclear inner membrane	nucleus
nuclear pore	nucleus

Table 4.1: A mapping of phrases to subcellular localizations for the nucleus localization label. The "subcellular localization" annotations of Swiss-Prot entries are searched for matches to any of these phrases. If a match is found the corresponding localization label is assigned to the protein.

in the final training data.

Subcellular localization categories are organism specific. For example, a bacterial cell does not have a nucleus, and an animal cell does not have a periplasmic space. To deal with this difference PA created a custom set of localizations for 5 different categories of biological cells: animal, green plant, fungi, gram negative bacteria and gram positive bacteria. This allows the classifiers trained on each organism to take advantage of the different properties that distinguish between the classes in different organisms. This also allows classifiers to make localization predictions that are as specific as possible, when specific localizations exist within a cell type. For example, if a protein localizes to the chloroplast, which exist only in green plants, we can call it a chloroplast protein using the green plant classifier. If we were to use a general subcellular localization classifier for all organisms we would have to generalize the chloroplast to the cytoplasmic class label (since the chloroplast is a cytoplasmic organelle). While calling a chloroplast protein cytoplasmic is not completely incorrect, it is not as correct as the class label chloroplast. For this dissertation, I chose to start with the animal data set because it is PA's largest, and because it has a diverse set of possible localizations.

PA's data sets have a separate "binary" training file for each class, resulting in a separate binary classifier for each class. Each query protein will be used as input for n separate classifiers, where n is the number of possible localizations. This determines whether the protein is or is not a member of each of the n classes. Each training instance appears once in every training file, but its class label changes between files. For example, in the nuclear data set a nuclear protein will appear with a positive label ("+1"), and non-nuclear proteins appear with a negative label ("−1"). The resulting classifier will return "+1" if it determines that a query protein is most likely nuclear, and "−1" otherwise. The PA training data includes 317 proteins that localize to more than one location. The binary schema of this data set allows proteins to appear with a positive label in more than one file. For example, a protein that is both cytoplasmic and peroxisomal will appear with the label "+1" in both the peroxisomal and cytoplasmic sets, and with the label "−1" in all other sets. Our data set has 7652 proteins across 9 classes (Table 4.2). To take advantage of the information in the abstracts of

Class Name	Number of Proteins	Number of Abstracts
cytoplasm	1664	4078
endoplasmic reticulum	310	666
extracellular	2704	5655
golgi	41	71
lysosome	129	599
mitochondrion	559	1228
nucleus	2445	5589
peroxisome	108	221
plasma membrane [a]	15	38
Total	**7652**	**17175**

[a] The plasma membrane class is very small, and performed poorly during experimentation. For this reason it has been removed from the results section.

Table 4.2: Summary of the PA Data Set. Totals are less than the sum of the rows because proteins may belong to more than one localization class.

proteins with multiple localizations, I use a one-against-all classification model, rather than a "single most confident class" approach.

4.2 Abstract Retrieval and Preprocessing

Now that a set of proteins with known localizations has been created, I gather each protein's abstracts and abstract titles (Figure 4.1, process a). I do not include full text because it can be difficult to obtain automatically and because previous research has shown that using full text does not improve F-measure [39]. Abstracts for each protein are retrieved using the PubMed IDs recorded in the Swiss-Prot database. PubMed (`http://www.pubmed.gov`) is a database of life science articles. It should be noted that more than one protein in Swiss-Prot may point to the same abstract in PubMed. Because the performance of the classifiers trained for this study is estimated using cross-validation (discussed in Section 4.4) it is important that the same abstract does not appear in both testing and training sets during any stage of cross-validation. To address this problem, all abstracts that appear more than once in the complete set of abstracts are removed. The distribution of the remaining abstracts among the 9 subcellular localization classes is shown in Table 4.2. For simplicity, the fact that an abstract may actually be discussing more than one protein is ignored. However, because I remove duplicate abstracts, many abstracts that discuss more than one protein are eliminated.

In Table 4.2 there are more abstracts than proteins because each protein may have more than one associated abstract. Because the plasma membrane class has so few proteins its performance is very poor (baseline F-measure of 0.000). For this reason I dropped the plasma membrane class from my experiments, though the plasma training instances remain as negative training data for the other 8

classes.

It is likely that not every abstract associated with a protein will discuss subcellular localization. However, because the Swiss-Prot entries for proteins in this data set all have subcellular annotations, some research must have been performed to ascertain localization. Thus it should be reported in at least one abstract. If the topics of the other abstracts are truly unrelated to localization, then their distribution of words may be the same for all localization classes. However, even if an abstract does not discuss localization directly, it may discuss some other property that is correlated with localization (e.g. function). In this case, the classifier will find terms that differentiate between the localization classes.

4.2.1 Preprocessing Text

In order to present the abstracts in a way that a classifier can understand, text must be broken down into individual words. I take a very simplistic approach to tokenization, where word boundaries are considered to be any form of white space, and all leading and trailing punctuation marks are stripped from the resulting words. In addition, because of the prevalence of hyphenated terms, words are also split on internal hyphens. Words are then stemmed using Porter's stemming algorithm [33] which strips the suffixes from words.

4.3 Leveraging the Gene Ontology

Three different data sets are made by processing the retrieved abstracts (Figure 4.1, process b). An example illustrating the three processing techniques is shown in Figure 4.3.

4.3.1 Baseline

In Data Set 1, words from all abstracts for a single protein are amalgamated into one "bag of words" that becomes the training instance representing the protein. The bag of words is transformed into a vector of ⟨word,value⟩ pairs. For this study I use two values for the feature vector: TFIDF and redundancy (Section 3.1.1). To review, TFIDF is defined as:

$$TFIDF(w_i) = f(w_i) * IDF = f(w_i) * log(\frac{n}{D(w_i)})$$

where $f(w_i)$ is the number of times word w_i appears in documents associated with a protein, n is the total number of training documents and $D(w_i)$ is the number of documents in the whole training set that contain the word w_i. Redundancy is defined as:

$$redundancy(w_i) = f(w_i) * r(w_i)$$

where $r(w_i)$ is:

$$r(w_i) = \log N + \sum_{j=1}^{N} \frac{f(w_i, d_j)}{f(w_i)} \log \frac{f(w_i, d_j)}{f(w_i)}$$

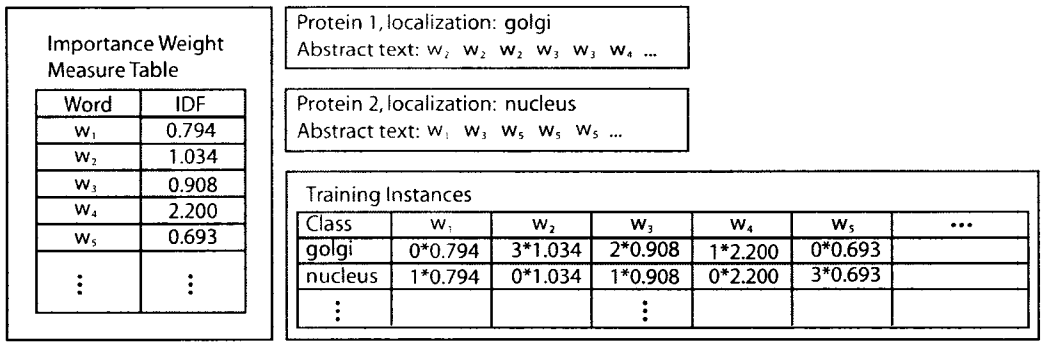

Figure 4.2: An example illustrating the creation of training instances using TFIDF. A table of importance weight measure values is created by counting the occurrences of words in the whole training set and using the counts in the IDF formula. A protein's abstracts is transformed into a training instance (feature vector) by multiplying the frequency of a given word in the abstract by its IDF value in the importance weight measure table

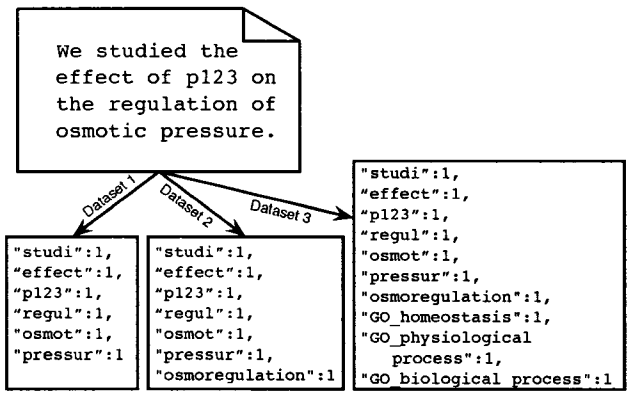

Figure 4.3: A sentence illustrating the three methods of abstract processing. Data Set 1 is the baseline, Data Set 2 incorporates synonym resolution and Data Set 3 incorporates synonym resolution and term generalization. Word counts are shown here for simplicity, though experiments use TFIDF and redundancy.

---

where $f(w_i, d_j)$ is the number of times word i appears in document j, $f(w_i)$ is the number of times word i appears in all documents and N is the number of documents.

Figure 4.2 shows how TFIDF is used to create training instances. First a table of IDF values is created in which each row corresponds to a word. A protein's abstracts are transformed into a training instance (feature vector) by multiplying the frequency of a given word in the abstract by its IDF value in the importance weight measure table. The same process is used to create training instance using the redundancy feature value, except that the importance weight measure table would contain $r(w_i)$ for each word w_i, instead of IDF.

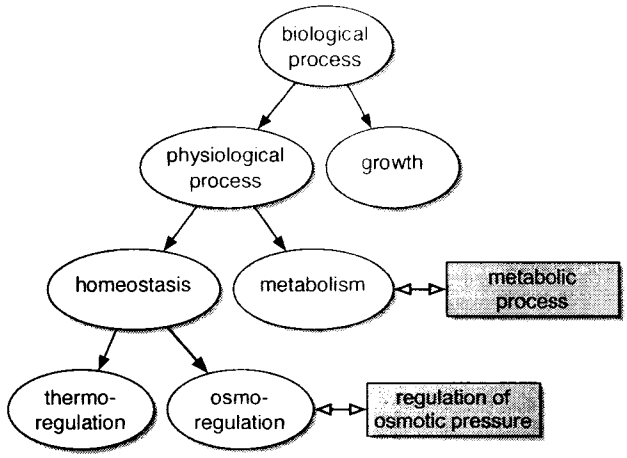

Figure 4.4: A sub-graph of the GO biological process hierarchy. GO nodes are shown as ovals, synonyms appear as grey rectangles. Arrows of the edges point to the more specific terms.

4.3.2 Synonym Resolution

The GO hierarchy can act as a thesaurus for biological both words and phrases. For example the GO encodes the fact that "metabolic process" is a synonym for "metabolism"(see Figure 4.4). Data Set 2 uses GO's "exact_synonym" field for synonym resolution and adds extra features to the vector of words from Data Set 1. I searched stemmed versions of the abstracts for matches to stemmed GO node names or synonyms. If a match is found, the GO node name (deemed the canonical representative for its set of synonyms) is associated with the abstract. In Figure 4.3 the phrase "regulation of osmotic pressure" appears in the text. A look-up in the GO hierarchy will indicate that this is an exact synonym of the GO node "osmoregulation". Therefore the term "osmoregulation" will be associated with the training instance. This approach combines the weight of several synonyms into one representative, allowing the SVM to more accurately model the author's intent, and identifies multi-word phrases that are otherwise lost during tokenization. Table 4.3 shows the increase in average number of positive features per training instance as a result of our synonym resolution technique.

4.3.3 Term Generalization

In order to express the relationships between terms, the GO hierarchy is organized in a directed acyclic graph (DAG). For example, "thermoregulation" is a type of "homeostasis", which is a "physiological process". This "is a" relationship is expressed as a series of parent-child relationships (see Figure 4.4). In Data Set 3, I use the GO for synonym resolution (as in Data Set 2) and I also use its hierarchical structure to generalize specific terms into broader concepts. For Data Set 3, if a GO node name (or synonym) is found in an abstract, all the names of ancestors of the matched term are

Class	Data Set 1	Data Set 2	Data Set 3
cytoplasm	166	177	203
endoplasmic reticulum	162	171	192
extracellular	148	155	171
golgi	131	143	172
lysosome	244	255	285
mitochondrion	155	163	186
nucleus	147	158	183
peroxisome	147	156	182
Overall Average	167	176	200

Table 4.3: The average number of features per training instance for 7 subcellular localization categories in animals. Data Set 1 is the baseline, Data Set 2 incorporates synonym resolution and Data Set 3 uses synonym resolution and term generalization.

included in the training instance along with word vectors from Data Set 2 (see Data Set 3 in Figure 4.3). These additional node names are prefixed with the string "GO_", which allows the SVM to differentiate between the case where a GO node name appears exactly in text and the case where a GO node name's child appeared in the text and the ancestor was added by generalization. Term generalization increases the average number of features per training instance (Table 4.3).

Term generalization gives the SVM algorithm the opportunity to learn correlations that exist between general terms and subcellular localization even if the general term itself never appears in an abstract and only the names of its more specific children occur. Without term generalization the SVM has no concept of the relationship between child and parent terms, nor between sibling terms. For some localization categories more general terms may be the most informative and in other cases specific terms may be best. Because my technique adds features to training instances and never removes any, the SVM can assign lower weights to the generalized terms in cases where the localization category is not well characterized by more general terms.

4.4 Evaluation

Each of the classifiers trained with the data sets described in this section are tested using 10-fold cross-validation. Each partition of the data is stratified so that it contains an approximately equal number of positive and negative training instances. The results of all 10 folds are combined and three composite measures of performance are calculated: precision, recall and F-measure.

Chapter 5

Results

This section covers the results of experiments outlined in Chapter 4. I explore the effects of using the GO hierarchy for synonym resolution and term generalization on PA's data set (described in Section 4.1) and on a data set created by the MultiLoc team [23] and used for text classification in their previous work [22]. Using 10-fold cross-validation, I calculated precision, recall and F-measure. T-tests (p=0.05) are used to test the statistical significance of the results using each of the 10 folds as a sample.

5.1 The PA Data Set

The first tests of the usefulness of term generalization and synonym resolution were performed using the PA data set. This set of experiments also served to compare two methods of importance weight measures outlined in Section 3.1.1: TFIDF and redundancy. I experimented with both un-normalized and L_2 normalized TFIDF and redundancy, but found that un-normalized representations obtained higher F-measures. Normalized results appear in Table 5.2 and Table 5.4. This section analyses only the un-normalized results that appear in Table 5.1 and Table 5.3.

The plasma membrane class, with only 15 training instances, produced very poor baseline classifiers. The plasma membrane baseline has F-measures of 0.000 for un-normalized redundancy, normalized TFIDF and normalized redundancy, and an F-measure of 0.095 for the un-normalized TFIDF classifier. Although techniques using the GO did yield improvements of as much as 0.095, the resulting classifiers were still of poor quality. Including improvements seen in the plasma membrane class inflates the average improvement for each technique. Since no technique can diminish a baseline of 0.000, and because our techniques failed to elevate the F-measure of the plasma membrane class above 0.100, the class was removed.

5.1.1 TFIDF

This section covers the results of using the TFIDF importance weight measure and compares the baseline classifier (Data Set 1) with classifiers trained on synonym resolution (Data Set 2) and the

combination of both synonym resolution and term generalization (Data Set 3). Overall, synonym resolution improves the F-measure by 0.015 to 0.789. In one class (golgi) the F-measure is decreased but the change is not statistically significant. Using the GO for synonym resolution significantly improves the F-measure for 6 of the 8 classes (see Table 5.1). The golgi class has less than 100 abstracts and less than 50 proteins, has a baseline F-measure of 0.500. Likely there is not enough data for synonym resolution to be of any use in this class, especially when the positive instances are split into groups for training and testing during cross validation. However, the peroxisome class, which is slightly larger (221 abstracts and 108 proteins), gains 0.033 in F-measure to 0.823. The lysosome class, with 129 proteins and just under 600 abstracts yields a 0.040 improvement in F-measure (to 0.787) with synonym resolution. I conjecture that there is a point between the size of the golgi class and the size of the peroxisome and lysosome classes where synonym resolution becomes useful.

Using the GO for both synonym resolution and term generalization brings further improvements to the classification of journal abstracts. Data Set 3 significantly improves the F-measure for 5 classes over the baseline. The average F-measure of Data Set 3 is 0.803, which is a significant improvement of 0.029 over the baseline. In 2 classes the F-measure for term generalization and synonym resolution is significantly better than that of synonym resolution alone, and in no case is it significantly worse. On average, the combination of term generalization and synonym resolution together is significantly better than synonym resolution alone. Based on these results it appears that using generalization and synonym resolution together is the best approach as it is never significantly worse, and sometimes is significantly better.

5.1.2 Redundancy

Redundancy, combined with synonym resolution, significantly increases the average F-measure of the 8 localization classifiers to 0.787, an improvement of 0.022 over baseline. The F-measure of the golgi class increases significantly. Two classes (nucleus and extracellular) have significantly decreased F-measure. Redundancy with term generalization and synonym resolution significantly increases the F-measure of 3 classes and significantly reduces the F-measure of two classes. The average F-measure of Data Set 3 is 0.801, a significant improvement of 0.038 over baseline. The same two classes (nucleus and extracellular) have significantly reduced F-measure compared to the baseline. However, on average, the F-measure of Data Sets 2 and 3 increase significantly over the baseline Data Set, and Data Set 3 significantly outperforms Data Set 2.

The two classes that perform significantly better without using the GO as a source of external information are also the largest in the PA training set, by far. Extracellular has 2704 proteins and nucleus has 2445 proteins (see Table 4.2). Because these classes are so large, there is a higher probability that the abstracts associated with the proteins contain most of the synonyms and specific terms that are generated in Data Sets 2 and 3. As synonym resolution and term generalization are

applied, the distribution of the added terms will be down-weighted by the redundancy importance weight measure as the distribution of the new terms is more uniform. Thus, the F-measures of the resulting classifiers do not improve (and actually decrease) when the GO is used as an information source.

Is it really just the size of the classes, and thus the already strong sampling of synonyms and specific terms, that makes Data Set 2 and 3 unsuccessful when paired with redundancy? To test this I created two new data sets: one that contains 25% of the nuclear training instances and the same negative training set, and one that contains 25% of the extracellular training instances. This brings the number of positive training instances for nuclear and extracellular to 611 and 676 respectively. The results of the 3 classifiers trained on this data appears in Table 5.5. As is expected, the classifiers do not perform as well as those trained with the entire data set. However, this exercise shows that synonym resolution and term generalization are once again useful techniques. The set of synonyms and specific terms has been reduced so the GO can be leveraged to compensate for the smaller sampling of synonyms and specific terms in the reduced training set.

Class	Data Set 1: Baseline			Data Set 2: Synonym Resolution				Data Set 3: Term Generalization			
	Precision	Recall	F-measure	Precision	Recall	F-Measure	Δ	Precision	Recall	F-Measure	Δ
cytoplasm	0.802	0.688	0.740 (±0.049)	0.817	0.706	**0.758** (±0.042)	+0.017	0.818	0.712	**0.761** (±0.042)	+0.021
endoplasmic reticulum	0.871	0.674	0.760 (±0.055)	0.869	0.706	**0.779** (±0.068)	+0.019	0.871	0.716	**0.786** (±0.072)	+0.026
extracellular	0.942	0.920	0.931 (±0.009)	0.944	0.927	**0.935** (±0.009)	+0.004	0.944	0.926	**0.935** (±0.010)	+0.004
golgi	0.581	0.439	0.500 (±0.153)	0.563	0.439	0.493 (±0.143)	-0.007	0.621	0.439	0.514 (±0.140)	+0.014
lysosome	0.822	0.682	0.746 (±0.107)	0.855	0.729	**0.787** (±0.100)	+0.041	0.870	0.775	**0.820*** (±0.089)	+0.074
mitochondrion	0.903	0.785	0.840 (±0.041)	0.903	0.800	**0.848** (±0.038)	+0.008	0.909	0.801	0.852 (±0.039)	+0.012
nucleus	0.887	0.882	0.885 (±0.014)	0.889	0.881	0.885 (±0.016)	+0.001	0.890	0.885	0.887 (±0.019)	+0.003
peroxisome	0.835	0.750	0.790 (±0.054)	0.851	0.796	**0.823** (±0.042)	+0.033	0.918	0.824	**0.868*** (±0.046)	+0.078
Average	0.830	0.728	0.774 (±0.022)	0.836	0.748	**0.789** (±0.019)	+0.015	0.855	0.760	**0.803*** (±0.019)	+0.029

Table 5.1: Precision, Recall and F-measures for stratified 10 fold cross-validation on three Data Sets created using the PA training set. The importance weight measure is un-normalized TFIDF. Results deemed significantly improved (p=0.05) over the baseline appear in **bold**, and those with an asterisk (*) are significantly better than both other data sets. Change in F-measure compared to baseline is shown for Data Sets 2 and 3. Standard deviation is shown in parentheses.

Class	Data Set 1: Baseline			Data Set 2: Synonym Resolution				Data Set 3: Term Generalization			
	Precision	Recall	F-measure	Precision	Recall	F-Measure	Δ	Precision	Recall	F-Measure	Δ
cytoplasm	0.885	0.632	0.737 (±0.040)	0.894	0.651	**0.753** (±0.036)	+0.016	0.893	0.655	**0.756** (±0.038)	+0.019
endoplasmic reticulum	0.961	0.642	0.770 (±0.040)	0.963	0.665	**0.786** (±0.041)	+0.016	0.958	0.665	0.785 (±0.037)	+0.015
extracellular	0.967	0.939	0.953 (±0.008)	0.967	0.940	0.954 (±0.009)	+0.001	0.966	0.936	0.951 (±0.010)	-0.002
golgi	0.875	0.171	0.286 (±0.237)	0.875	0.171	0.286 (±0.237)	0.000	1.000	0.171	0.292 (±0.281)	+0.006
lysosome	0.944	0.659	0.776 (±0.103)	0.956	0.674	0.791 (±0.099)	+0.015	0.958	0.705	**0.813** (±0.083)	+0.036
mitochondrion	0.981	0.723	0.832 (±0.046)	0.979	0.733	0.838 (±0.049)	+0.006	0.976	0.721	0.829 (±0.051)	-0.003
nucleus	0.924	0.881	0.902 (±0.014)	0.927	0.884	0.905 (±0.012)	+0.003	0.926	0.879	0.902 (±0.014)	0.000
peroxisome	0.964	0.750	0.844 (±0.125)	0.965	0.769	0.856 (±0.114)	+0.012	0.988	0.741	0.847 (±0.087)	+0.003
Average	0.938	0.674	0.762 (±0.025)	0.941	0.686	**0.771** (±0.024)	+0.009	0.958	0.684	0.772 (±0.036)	+0.009

Table 5.2: Precision, Recall and F-measures for stratified 10 fold cross-validation on three Data Sets made with the PA training set. The importance weight measure is L_2 normalized TFIDF. Results deemed significantly improved (p=0.05) over the baseline appear in **bold**, and those with an asterisk (*) are significantly better than both other data sets. Change in F-measure compared to baseline is shown for Data Sets 2 and 3. Standard deviation is shown in parentheses.

	Data Set 1: Baseline			Data Set 2: Synonym Resolution				Data Set 3: Term Generalization			
Class	Precision	Recall	F-measure	Precision	Recall	F-Measure	Δ	Precision	Recall	F-Measure	Δ
cytoplasm	0.887	0.635	0.740 (±0.038)	0.815	0.697	0.752 (±0.044)	+0.011	0.818	0.710	**0.760** (±0.040)	+0.020
endoplasmic reticulum	0.961	0.642	0.770 (±0.041)	0.865	0.700	0.774 (±0.068)	+0.004	0.886	0.726	**0.798** (±0.069)	+0.028
extracellular	0.970	0.939	**0.954*** (±0.008)	0.942	0.928	0.935 (±0.007)	-0.019	0.944	0.930	0.937 (±0.009)	-0.017
golgi	0.875	0.171	0.286 (±0.237)	0.563	0.439	**0.493** (±0.143)	+0.207	0.600	0.439	**0.507** (±0.147)	+0.221
lysosome	0.944	0.651	0.771 (±0.103)	0.826	0.736	0.779 (±0.107)	+0.008	0.847	0.775	0.810 (±0.090)	+0.039
mitochondrion	0.983	0.733	0.840 (±0.052)	0.901	0.801	0.848 (±0.043)	+0.008	0.903	0.800	0.848 (±0.038)	+0.008
nucleus	0.925	0.881	**0.903*** (±0.015)	0.894	0.885	0.889 (±0.015)	-0.013	0.894	0.885	0.889 (±0.019)	-0.013
peroxisome	0.964	0.750	0.844 (±0.121)	0.860	0.796	0.827 (±0.037)	-0.017	0.907	0.815	0.859 (±0.048)	+0.015
Average	0.939	0.675	0.763 (±0.024)	0.833	0.748	**0.787** (±0.022)	+0.024	0.850	0.760	**0.801*** (±0.020)	+0.038

Table 5.3: Precision, Recall and F-measures for stratified 10 fold cross-validation on three Data Sets created using PA's training data. The importance weight measure is un-normalized redundancy. Results deemed significantly improved (p=0.05) over the baseline appear in **bold**, and those with an asterisk (*) are significantly better than both other data sets. Change in F-measure compared to baseline is shown for Data Sets 2 and 3. Standard deviation is shown in parentheses.

Class	Data Set 1: Baseline			Data Set 2: Synonym Resolution				Data Set 3: Term Generalization			
	Precision	Recall	F-measure	Precision	Recall	F-Measure	Δ	Precision	Recall	F-Measure	Δ
cytoplasm	0.887	0.635	0.740 (±0.038)	0.892	0.650	**0.752** (±0.039)	+0.012	0.895	0.658	**0.759** (±0.042)	+0.018
endoplasmic reticulum	0.961	0.642	0.770 (±0.041)	0.963	0.665	**0.786** (±0.041)	+0.016	0.958	0.665	**0.785** (±0.037)	+0.015
extracellular	0.970	0.939	0.954 (±0.008)	0.969	0.939	0.954 (±0.008)	0.000	0.965	0.939	0.952 (±0.009)	-0.002
golgi	0.875	0.171	0.286 (±0.237)	0.875	0.171	0.286 (±0.237)	0.000	1.000	0.171	0.292 (±0.281)	+0.006
lysosome	0.944	0.651	0.771 (±0.103)	0.957	0.682	**0.796** (±0.092)	+0.026	0.958	0.705	**0.813** (±0.83)	+0.042
mitochondrion	0.983	0.733	0.840 (±0.052)	0.979	0.737	0.841 (±0.052)	+0.001	0.976	0.728	0.834 (±0.053)	-0.006
nucleus	0.925	0.881	0.903 (±0.015)	0.925	0.883	0.903 (±0.014)	0.000	0.927	0.879	0.902 (±0.015)	0.000
peroxisome	0.964	0.750	0.844 (±0.121)	0.965	0.769	0.856 (±0.110)	+0.012	0.988	0.741	0.847 (±0.087)	+0.003
Average	0.939	0.675	0.763 (±0.024)	0.940	0.687	**0.772** (±0.025)	+0.008	0.958	0.686	0.773 (±0.036)	+0.009

Table 5.4: Precision, Recall and F-measures for stratified 10 fold cross-validation three Data Sets created using PA's training data. The importance weight measure is L_2normalized redundancy. Results deemed significantly improved (p=0.05) over the baseline appear in **bold**, and those with an asterisk (*) are significantly better than both other data sets. Change in F-measure compared to baseline is shown for Data Sets 2 and 3. Standard deviation is shown in parentheses.

Class	Data Set 1: Baseline			Data Set 2: Synonym Resolution				Data Set 3: Term Generalization			
	Precision	Recall	F-measure	Precision	Recall	F-Measure	Δ	Precision	Recall	F-Measure	Δ
extracellular	0.896	0.785	0.836 (±0.031)	0.910	0.792	**0.847** (±0.034)	+0.011	0.918	0.796	**0.853** (±0.029)	+0.017
nucleus	0.869	0.740	0.799 (±0.042)	0.883	0.755	**0.814** (±0.039)	+0.014	0.879	0.764	**0.818** (±0.033)	+0.019

Table 5.5: Precision, Recall and F-measures for stratified 10 fold cross-validation of three Data Sets created using the two largest classes of PA's training set. The importance weight measure is un-normalized redundancy. Data sets were altered to contain only one quarter of the positive training instances and the same number of negative training instances. Results are shown for the two largest classes in the PA training set. Results deemed significantly improved (p=0.05) over the baseline appear in **bold**, and those with an asterisk (*) are significantly better than both other data sets. Change in F-measure compared to baseline is shown for Data Sets 2 and 3. Standard deviation is shown in parentheses.

Class Name	Best Technique TFIDF	F-Measure	Best Technique Redundancy	F-Measure
cytoplasm	Data Set 3*	0.761	Data Set 3+	0.760
endoplasmic reticulum	Data Set 3*	0.786	Data Set 3+	**0.798**
extracellular	Data Set 2*	0.935	Data Set 1**	**0.954**
golgi	Data Set 3	0.514	Data Set 3*	0.507
lysosome	Data Set 3**	**0.820**	Data Set 3	0.810
mitochondrion	Data Set 2+	0.848	Data Set 3	0.848
nucleus	Data Set 3	0.887	Data Set 1**	**0.903**
peroxisome	Data Set 3**	0.868	Data Set 3	0.859

Table 5.6: Comparison of TFIDF and redundancy as importance weight measures for subcellular classification of the PA Set. Data Sets followed by one asterisk (*) or one plus (+) signify that Data Set was significantly better than one other Data Set in the corresponding experiment (Tables 5.1 and 5.3), two asterisks (**) signify that the Data Set was significantly better than both other Data Sets in the corresponding experiment. F-measures for the techniques where TFIDF or redundancy is significantly better are shown in bold.

5.1.3 TFIDF vs. Redundancy

The average F-measures of the classifiers trained on the PA data set that use TFIDF as the importance weight measure are very close to those that use redundancy as the importance weight measure. To fairly compare the two importance weight measures I compared the results on a class-by-class basis. In order to compare importance weight measures I chose the best classifier across the three abstract processing techniques for each class and for each importance weight measure (TFIDF and redundancy). Best was defined for each class as follows:

- If a technique was significantly better than all other techniques it was chosen (denoted ** in Table 5.6 and Table 5.14).
- If technique A and technique B are both significantly better than technique C, but there is no significant difference between A and B the technique with the higher F-measure was chosen. If F-measures were the same, the technique with a lower standard deviation was chosen (denoted * in Table 5.6 and Table 5.14).
- If a technique A was significantly better than technique B and there was no significant difference between any other pair of techniques, technique A was chosen (denoted + in Table 5.6 and Table 5.14).
- If no techniques are significantly better I chose the technique with the highest F-measure. If F-measures were the same, the technique with a lower standard deviation was chosen (no marks in Table 5.6 and Table 5.14).

The cross-validation results of the best classifiers are then compared using a t-test. Results are appear in Table 5.6.

The two largest classes in the PA data set, nucleus and extracellular, create a baseline redundancy classifier that significantly outperforms every other combination of technique and importance weight measure. In one other case, endoplasmic reticulum, redundancy is the best importance weight measure, when combined with Data Set 3. For the lysosome class TFIDF and Data Set 3 is significantly better than every other combination of technique and importance weight measure. In all other classes (cytoplasm, golgi, mitochondrion, peroxisome) the difference between the best classifiers is not significant.

ANOVA tests

Performing significance tests several times reduces the reliability of their predictions. Because it is necessary to perform several pairwise significance tests to compare all techniques, the power of the tests has degraded. For this reason I perform ANOVA (Analysis of Variance) [36] for the 6 classifiers trained for each class (TFIDF with Data Sets 1, 2 and 3 and redundancy with Data Sets 1, 2 and 3). This results of ANOVA for the classifiers trained with the PA Data Set shows that redundancy and baseline is best for the extracellular class. Although there was no significant difference in t-test between the 3 techniques using TFIDF, ANOVA shows that Data Set 3 and TFIDF is significantly better than all other techniques and importance weight measures. Overall, ANOVA shows that the average performance of classifiers using Data Set 3 and TFIDF is significantly better than all other combinations.

5.2 Case Study

Experiments thus far have shown that when using TFIDF, synonym resolution and term generalization together give significant gains. However, when using redundancy, the GO can actually hurt a classifier's performance. What, exactly, is happening to the weights of features within the SVM classifier? Let us look at two examples: one where Data Set 3 significantly outperforms Data Set 1 when using TFIDF, and one where Data Set 1 significantly outperforms Data Set 3 when using redundancy as the importance weight measure.

5.2.1 Example: TFIDF and a protein from the lysosome

Let us start with a case where Data Set 3 is significantly better than Data Set 1. Consider the following lysosomal protein (accession number P11117) from the Swiss-Prot database. In Swiss-Prot this protein is given the following subcellular localization annotation: "Lysosome". The abstract for P11117 ([18]) appears below. The most heavily weighted words appear in bold in the following abstract text. Words that are underlined generate GO terms that are in the top 5 highest weighted words for the classifier trained on Data Set 3.

Swiss-Prot accession number P11117

Word	weight
lysosom	0.078937
phosphatas	0.024493
acid	0.010438
human	0.007127
structur	0.000623

Table 5.7: The top five most heavily weighted words from the abstract related to protein P11117. Weights are determined by an SVM trained on Data Set 1 (baseline) with TFIDF. These words are shown in bold in the abstract of P11117.

Word	weight
lysosom	0.033768
go_lysosom	0.031121
go_lyt vacuol	0.030902
go_vacuol	0.030858
phosphatas	0.014357

Table 5.8: The top five most heavily weighted words and GO terms from the abstract related to protein P11117. Weights are determined by an SVM trained on Data Set 3 (synonym resolution and term generalization) with TFIDF. Words or phrases that generate GO terms that appear in this table are shown underlined in P11117's abstract. Words in this table that are not generated using the GO are shown in bold in the abstract of P11117.

Title 1: ***Structure** of the **human** lysosomal **acid phosphatase** gene.*

Abstract Body 1: We have isolated a 12-kb genomic clone, which encodes **human** lysosomal **acid phosphatase** (LAP), a lysosomal membrane glycoprotein. The **human** LAP gene has a size of about 9 kb and contains 11 exons (83-947 bp in size). The signal sequence and the first eight amino **acids** of the LAP protein are encoded by exon 1, the remaining luminal domain by exons 2-10 and the transmembrane and cytoplasmic domains, as well as the 3'-untranslated region, by exon 11. The sequence of the LAP gene confirmed the sequence deduced from the cDNA clone except for nucleotide 1917 in the 3'-untranslated region, where T is changed to C. The 5'-flanking sequence shows promoter activity, as analysed by coupling to bacterial chloramphenicol acetyltransferase. S1-nuclease-protection and primer-extension analysis demonstrate transcription initiation at multiple sites clustering within 23 bp upstream of the translation-initiation codon. Sequences characteristic for promoter regions like TATA-box and CAAT-box sequences could not be identified at typical positions. The absence of these sequences, the high GC content (63.5%), two GC boxes and a region complying with the properties of a CpG island, indicate that LAP is a housekeeping gene.

The training instance created from P11117 was misclassified during cross-validation for the baseline classifier, but was correctly classified during cross-validation for Data Set 3. When this

abstract is used as input for the SVM trained on all of Data Set 1 using TFIDF, the five stemmed words in Table 5.7 have the highest weight. The first three words are intuitive; "lysosom" is the stemmed version of lysosome and lysosomal, and both phosphatase (stemmed to "phosphatas") and acids are found in lysosomes. However, during cross-validation, this protein is classified incorrectly as "not lysosomal" because there was not enough evidence to call it a lysosomal protein. Part of the problem is that two words, "human" and "structur" (stemmed version of structure), end up in the top 5 words even though they have low weights and have little to do with the lysosome. This is indicative of the fact that there are not enough positive features for the SVM to predict the lysosome class. In order to turn this not lysosome prediction into a lysosome prediction we have to increase the weight of the positive words relative to the negative words, or we have to generate words that have positive weight.

When a lysosome classifier trained on Data Set 3 makes a prediction on the abstract of protein P11117 the classifier returns a positive prediction. The five features with the highest weight are shown in Table 5.8. In this example, 3 terms created using the GO hierarchy appear in the top 5 terms: "lysosome" (parent of lysosomal membrane which appears in the abstract), "lytic vacuole" (parent of lysosome) and "vacoule" (parent of lytic vacuole). Now that we have generated additional features with positive weights there is enough evidence for this protein to be correctly classified as lysosomal during cross validation.

5.2.2 Example: Redundancy and a protein from the nucleus

Let us turn now to a case where Data Set 1 is significantly better than Data Set 3. Consider the following nuclear protein with Swiss-Prot accession number O95707. This protein is annotated with Subcellular location: "Nucleus; nucleolus", and has two associated abstracts ([26, 44]). The most heavily weighted words appear in bold in the following abstract text. Words that are underlined generate GO terms that are in the top 5 highest weighted words for the classifier trained on Data Set 3.

> **Swiss-Prot accession number O95707**
>
> Title 1: *Rpp14 and Rpp29, two protein subunits of human ribonuclease P.*
>
> Abstract Body 1: In **HeLa** cells, the tRNA processing enzyme ribonuclease P (RNase P) consists of an **RNA** molecule associated with at least eight protein subunits, hPop1, Rpp14, Rpp20, Rpp25, Rpp29, Rpp30, Rpp38, and Rpp40. Five of these proteins (hPop1p, Rpp20, Rpp30, Rpp38, and Rpp40) have been partially characterized. Here we report on the cDNA cloning and immunobiochemical analysis of Rpp14 and Rpp29. Polyclonal rabbit antibodies raised against recombinant Rpp14 and Rpp29 recognize their corresponding antigens in **HeLa** cells and precipitate catalytically active RNase P. Rpp29 shows 23% identity with Pop4p, a subunit of **yeast nuclear** RNase P and the

ribosomal **RNA** processing enzyme RNase MRP. Rpp14, by contrast, exhibits no significant homology to any known **yeast** gene. Thus, human RNase P differs in the details of its protein composition, and perhaps in the functions of some of these proteins, from the **yeast** enzyme.

Title 2: *hPop4: a new protein subunit of the human RNase MRP and RNase P ribonucleoprotein* ***complexes.***

Abstract Body 2: RNase MRP is a ribonucleoprotein particle involved in the processing of pre-rRNA. The RNase MRP particle is structurally highly related to the RNase P particle, which is involved in pre-tRNA processing. Their **RNA** components fold into a similar secondary structure and they share several protein subunits. We have identified and characterised human and mouse cDNAs that encode proteins homologous to yPop4p, a protein subunit of both the **yeast** RNase MRP and RNase P **complexes**. The human Pop4 cDNA encodes a highly basic protein of 220 amino acids. Transfection experiments with epitope-tagged hPop4 protein indicated that hPop4 is localised in the nucleus and accumulates in the nucleolus. Immunoprecipitation assays using extracts from transfected cells expressing epitope-tagged hPop4 revealed that this protein is associated with both the human RNase MRP and RNase P particles. Polyclonal rabbit antibodies raised against recombinant hPop4 recognised a 30 kDa protein total **HeLa** cell extracts and specifically co-immunoprecipitated the **RNA** components of the RNase MRP and RNase P **complexes**. Finally we showed that anti-hPop4 immunoprecipitates possess RNase P enzymatic activity. Taken together, these data show that we have identified a protein that represents the human counterpart of the **yeast** Pop4p protein.

The top five highest-weighted words from the abstracts associated with O95707 for the baseline classifier using redundancy are shown in Table 5.9. In this case, the baseline classifier gives very high weight to the word "nuclear", which is the adjective form of the class name nucleus. Nuclear is understandably a very good indicator that the protein's localization in in the nucleus. RNA and HeLa are also intuitive indicators of the nucleus class. RNA is manufactured and processed in the nucleus. HeLa is a type of cell often used in cancer research, and much of cancer research involves processes that occur in the nucleus.

Table 5.10 shows the top five words with the highest weight for the SVM classifier trained on Data Set 3 using redundancy. Nuclear, yeast and HeLa appear again in the top 5 words. RNA has fallen out of the top 5 and been replaced with the GO term "organelle" which has nucleus, nucleolus and ribosome as descendants. Unfortunately, in the GO, nucleus has siblings mitochondrion and golgi apparatus. These two sibling terms will also expand and produce the GO term "organelle" during generalization. Thus, organelle is a bad word to weight heavily during classification since it differentiates between cytoplasm and not cytoplasm classes, but not between cytoplasmic classes

Word	weight
nuclear	5.682055
yeast	1.643581
rna	1.571278
complex	1.407400
hela	1.020248

Table 5.9: The top five most heavily weighted words from abstracts related to protein O95707. Weights are determined by an SVM trained on Data Set 1 (baseline) with TFIDF. These words are shown in bold in the abstract of O95707.

Word	weight
nuclear	0.059546
yeast	0.029832
go_organel	0.022970
complex	0.019493
hela	0.016555

Table 5.10: The top five most heavily weighted words and GO terms from abstracts related to protein O95707. Weights are determined by an SVM trained on Data Set 3 (synonym resolution and term generalization) with TFIDF. Words or phrases that generate GO terms that appear in this table are shown underlined in O95707's abstract. Words in this table that are not generated using the GO are shown in bold in the abstract of O95707.

(nuclear, mitochondrion, golgi, etc.).

The word RNA, which ranked highly in the baseline classifier, has fallen to 12th place, with a weight of 0.010837. In 11th place, ranking higher than RNA, is the GO term "cellular process", which is actually the root of a tree in the GO hierarchy, meaning that it will be associated with any abstract that mentions any term in the cellular process hierarchy. "Cellular process" is likely a horrible indicator of any class, and appears with abundance in abstracts from all classes (it will be produced 5 times when term generalization is applied to the abstracts associated with the nuclear protein O95707 and 6 times for the abstracts of the lysosome protein in Example 1). For this study, I had decided to leave the root nodes in the training set, assuming the SVM would learn their utility, or lack thereof. Future work will explore leaving out the root nodes. More generally, a learning task should be developed to discover the level of generalization that is optimal for each localization class. In addition, feature selection (the process of identifying before learning which features are likely good indicators of class) could be used to remove GO nodes names and other words which are poor predictors of class.

The redundancy feature weight measure may actually be working against the features generated by synonym resolution and term generalization. Note that the terms generated by synonym resolution and term generalization will be at least as uniformly distributed as the synonyms or specific terms in the abstracts that generate them. Because redundancy down-weights uniformly distributed terms, the generated terms will have less weight in the training instances. These additional features create noise in the training instances from which the SVM cannot recover.

Class Name	Number of Proteins	Proteins with Abstracts	Number of Abstracts
cytoplasmic	1411	1344	4962
ER	198	187	617
extracellular	843	825	3373
Golgi	150	150	456
lysosomal	103	102	745
mitochondrial	510	496	1888
nuclear	837	788	2564
peroxisomal	157	147	437
plasma membrane	1238	1198	4661
Total	**5447**	**5237**	**19703**

Table 5.11: Summary of distribution of proteins and abstracts in the MultiLoc Data Set. Proteins without abstracts were not used to train or test text classifiers.

5.3 MultiLoc Data Set

The MultiLoc data set has been used to test text classification as input to a larger subcellular localization classifier [22] . The MultiLoc data set was created by searching for key phrases in the Subcellular localization and Feature fields of the Swiss-Prot database. This resulted in a data set of 9,761 proteins, which was reduced by removing sequences until no pair of sequences shared $> 80\%$ sequence similarity. The final MultiLoc data set has 5,447 proteins, 5,237 with abstracts. Although the first step in the creation process for the MultiLoc data set is very similar to the process for creating the PA data set, the distribution of proteins amongst the classes is very different (see Table 5.11). For example, the plasma membrane class in the MultiLoc data set is the second largest class and contains 23% of all proteins in the data set. The plasma membrane class in the PA data set is the smallest class and makes up less than 0.2% of the data set. For some localizations, removing similar proteins contributes to the difference in size and distribution of the classes.

5.3.1 TFIDF

Results for 10-fold cross validation of the MultiLoc data set using TFIDF as the importance weight measure appear in Table 5.12. Using TFIDF gives an average baseline F-measure of 0.749. When synonym resolution is applied, this average rises significantly to 0.755. Three of the nine classes improve significantly with this measure: cytoplasm, mitochondrial and plasma membrane. The extracellular class loses 0.001 in F-measure, but this is not significant.

Using term generalization and synonym resolution the average F-measure improves significantly over the baseline to 0.757 but is not significantly better than the average of synonym resolution alone. The cytoplasm and plasma membrane classes have significantly improved F-measure over the baseline. Synonym resolution and term generalization used together is significantly better than

synonym resolution and the baseline for the plasma membrane class.

5.3.2 Redundancy

Results for experiments using the MultiLoc data set and redundancy as the importance weight measure appear in Table 5.13. The baseline average F-measure using redundancy as the feature value is 0.742. Using synonym resolution significantly increases the average F-measure to 0.749. Four classes (cytoplasm, lysosomal, nuclear and plasma membrane) achieve significantly higher F-measures using synonym resolution.

Term generalization and synonym resolution together significantly increases the F-measure of four classes over the baseline (cytoplasm, extracellular, nuclear and plasma membrane). The average F-measure for this technique is 0.751, which is significantly higher than baseline. The F-measure for the golgi class drops, but the change is not significant.

Significant losses were seen in PA Data Sets 2 and 3 when redundancy was used as the importance weight measure for large classes. We do not see the same decreases in performance using redundancy on the largest classes (cytoplasmic and plasma membrane) of the MultiLoc data. The large classes both produce significantly better classifiers than baseline using Data Set 2 and 3. Although the largest classes are large, they are still only about half the size of the two largest classes in the PA data, probably due to the removal of similar sequences in the MultiLoc data. Because the classes are large, but not too large, techniques which use the GO hierarchy are still of use.

	Data Set 1: Baseline			Data Set 2: Synonym Resolution				Data Set 3: Term Generalization			
Class	Precision	Recall	F-measure	Precision	Recall	F-Measure	Δ	Precision	Recall	F-Measure	Δ
cytoplasm	0.759	0.711	0.734 (±0.027)	0.762	0.722	**0.742** (±0.024)	+0.008	0.766	0.727	**0.746** (±0.020)	+0.012
ER	0.572	0.594	0.583 (±0.084)	0.575	0.594	0.584 (±0.093)	+0.002	0.568	0.604	0.585 (±0.089)	+0.003
extracellular	0.897	0.819	0.856 (±0.031)	0.897	0.817	0.855 (±0.030)	-0.001	0.903	0.825	0.863 (±0.029)	+0.006
golgi	0.881	0.740	0.804 (±0.062)	0.883	0.753	0.813 (±0.069)	+0.009	0.864	0.760	0.809 (±0.068)	+0.004
lysosomal	0.887	0.618	0.728 (±0.115)	0.878	0.637	0.739 (±0.110)	+0.010	0.857	0.647	0.737 (±0.100)	+0.009
mitochondrial	0.685	0.696	0.690 (±0.049)	0.697	0.700	**0.698** (±0.047)	+0.008	0.699	0.692	0.695 (±0.050)	+0.005
nuclear	0.791	0.778	0.784 (±0.037)	0.797	0.783	0.790 (±0.034)	+0.006	0.791	0.786	0.788 (±0.037)	+0.004
peroxisomal	0.769	0.680	0.722 (±0.072)	0.779	0.694	0.734 (±0.070)	+0.012	0.776	0.707	0.740 (±0.081)	+0.018
plasma membrane	0.828	0.842	0.835 (±0.025)	0.837	0.846	**0.841** (±0.024)	+0.006	0.845	0.855	**0.850*** (±0.029)	+0.015
Average	0.785	0.720	0.749 (±0.020)	0.790	0.727	**0.755** (±0.019)	+0.007	0.785	0.734	**0.757** (±0.021)	+0.008

Table 5.12: Precision, Recall and F-measures for stratified 10 fold cross-validation of three Data Sets using un-normalized TFIDF and the MultiLoc data set. Results deemed significantly improved (p=0.05) over the baseline appear in **bold**, and those with an asterisk (*) are significantly better than both other data sets. Change in F-measure compared to baseline is shown for Data Sets 2 and 3. Standard deviation is shown in parentheses.

Class	Data Set 1: Baseline			Data Set 2: Synonym Resolution				Data Set 3: Term Generalization			
	Precision	Recall	F-measure	Precision	Recall	F-Measure	Δ	Precision	Recall	F-Measure	Δ
cytoplasm	0.746	0.700	0.722 (±0.026)	0.752	0.710	**0.730** (±0.025)	+0.008	0.755	0.719	**0.737** (±0.019)	+0.015
ER	0.562	0.604	0.582 (±0.090)	0.562	0.610	0.585 (±0.094)	+0.002	0.569	0.620	0.593 (±0.093)	+0.011
extracellular	0.896	0.802	0.847 (±0.035)	0.897	0.804	0.848 (±0.035)	+0.001	0.900	0.816	**0.856** (±0.030)	+0.009
golgi	0.874	0.740	0.801 (±0.063)	0.874	0.740	0.801 (±0.063)	0.000	0.867	0.740	0.799 (±0.066)	-0.003
lysosomal	0.855	0.637	0.730 (±0.117)	0.870	0.657	**0.749** (±0.127)	+0.018	0.848	0.657	0.740 (±0.115)	+0.010
mitochondrial	0.677	0.702	0.689 (±0.051)	0.678	0.696	0.687 (±0.056)	-0.003	0.690	0.690	0.690 (±0.051)	+0.000
nuclear	0.766	0.765	0.766 (±0.040)	0.780	0.770	**0.775** (±0.036)	+0.010	0.779	0.778	**0.778** (±0.032)	+0.013
peroxisomal	0.739	0.694	0.716 (±0.072)	0.746	0.701	0.723 (±0.076)	+0.007	0.738	0.707	0.722 (±0.084)	+0.006
plasma membrane	0.816	0.837	0.827 (±0.031)	0.830	0.848	**0.839** (±0.032)	+0.012	0.833	0.857	**0.845** (±0.030)	+0.018
Average	0.770	0.720	0.742 (±0.021)	0.777	0.726	**0.749** (±0.021)	+0.006	0.775	0.732	**0.751** (±0.023)	+0.009

Table 5.13: Precision, Recall and F-measures for stratified 10 fold cross-validation of three Data Sets using un-normalized redundancy and the MultiLoc data set. Results deemed significantly improved (p=0.05) over the baseline appear in **bold**, and those with an asterisk (*) are significantly better than both other data sets. Change in F-measure compared to baseline is shown for Data Sets 2 and 3. Standard deviation is shown in parentheses

Class Name	Best Technique TFIDF	F-Measure	Best Technique Redundancy	F-Measure
cytoplasm	Data Set 3*	**0.746**	Data Set 3*	0.737
ER	Data Set 3	0.585	Data Set 3	0.593
extracellular	Data Set 3	0.863	Data Set 3+	0.856
Golgi	Data Set 2	0.813	Data Set 2	0.801
lysosomal	Data Set 2	0.739	Data Set 2+	0.749
mitochondrial	Data Set 2+	**0.698**	Data Set 3	0.690
nuclear	Data Set 2	**0.790**	Data Set 3*	0.778
peroxisomal	Data Set 3	**0.740**	Data Set 2	0.723
plasma membrane	Data Set 3**	**0.850**	Data Set 3*	0.845

Table 5.14: Comparison of TFIDF and redundancy as importance weight measures for subcellular classification of the MultiLoc Set. Data Sets followed by one asterisk (*) or one plus (+) signify that Data Set was significantly better than one other Data Set in the corresponding experiment (Tables 5.1 and 5.3), two asterisks (**) signify that the Data Set was significantly better than both other Data Sets in the corresponding experiment. F-measures for the techniques where TFIDF or redundancy is significantly better are shown in bold.

5.3.3 TFIDF vs. Redundancy

On average, the classifiers trained on the MultiLoc data set that use TFIDF as the importance weight measure out-perform classifiers that use redundancy. Averages can be misleading, so again I explored the classifiers on a class-by-class basis. To compare TFIDF and redundancy, I compared the cross validation results for the best classifier for each of the classes and each of TFIDF and redundancy. Again, the best classifier is chosen as outlined in Section 5.1.3.

Table 5.14 shows that the best results for TFIDF are significantly better than the best results for redundancy for the classes cytoplasm, mitochondrial, nuclear, peroxisomal and plasma membrane. Three out of these five TFIDF classifiers use Data Set 3 and the other two use Data Set 2. For three of these classes (cytoplasm, mitochondrial and plasma membrane) the selected Data Set was better than baseline, and in the plasma membrane class the selected data set, Data Set 3, was better than Data Set 1 and 2. This makes a strong case for using TFIDF as the importance weight measure over redundancy.

ANOVA tests

Again, because so many pairwise tests were performed, ANOVA tests are needed. ANOVA tests show no significant difference between any importance weight measure and technique for the MulitLoc Data Set.

5.3.4 Comparison to MultiLoc's classifiers

A comparison of the results of my best text classification technique (Opt-Text) and MultiLoc's text classification and combined techniques is shown in Table 5.15. MultiLoc's text classifier uses an

Class Name	MultiLoc Text only	MultiLoc Combined	Opt-Text	Opt-Text minus MultiLoc
cytoplasm	0.614	0.868	0.746	+0.132
ER	0.582	0.737	0.593	+0.011
extracellular	0.770	0.880	0.863	+0.093
Golgi	0.546	0.744	0.813	+0.267
lysosomal	0.449	0.671	0.749	+0.300
mitochondrial	0.795	0.920	0.698	-0.097
nuclear	0.770	0.859	0.790	+0.020
peroxisomal	0.729	0.771	0.740	+0.011
plasma membrane	0.851	0.897	0.850	-0.001

Table 5.15: Comparison of MulitLoc's techniques and the techniques presented in this dissertation (Opt-Text) on the MultiLoc Data Set. The rightmost column is calculated by subtracting the F-measure of the MultiLoc text only classifier from the F-measure of the Opt-Text classifier.

SVM. A Z-score is used to determine the most statistically significant words, which are then used to represent the text of the abstracts. The non-standard importance weight measure of each distinguishing term is the number of times a distinguishing term occurs in a protein's abstracts, divided by the number of times any distinguishing term appears in the protein's abstracts. As discussed in Section 2.2.1, MultiLoc's larger system is a set of classifiers which produce output based on the protein's amino acid sequence. These sub-classifiers are combined with the classification of MultiLoc's text-classifier using one super SVM which makes a final subcellular localization prediction. MultiLoc's combined results (Table 5.15) are the results using these sequence based classifiers as well as their text-based classifier in the final larger system of classifiers. MultiLoc used Swiss-Prot release 42.0 to produce their results and I used Swiss-Prot 50.0 to retrieve abstracts for my experiments.

My system never out-performs the MultiLoc combined classifier because MultiLoc's combined classifier has so much more information available to it. Based on F-measure, Opt-Text outperforms the MultiLoc text classifier in 7 of the 9 classes, sometimes by as much as 0.300. The average improvement of Opt-Text is 0.082. This improvement in F-measure may be due to the change in Swiss-Prot, as later releases have more textual data. The MultiLoc classifier that incorporates the MultiLoc text classifier saw significant improvements when they incorporated their text classifiers. One would assume the improvement of the combined classifier would be even greater if the underlying text classifier was improved. There is no reason why the techniques presented in this dissertation could not be incorporated into MultiLoc's text classifier and then used as part of MultiLoc's combined classifier.

5.4 Discussion

Using the GO hierarchy for synonym resolution and term generalization results in a significant gain when TFIDF is used as the importance weight measure, especially for smaller classes. However, for very large classes, Data Sets that use redundancy as the importance weight measure and do not make use of the GO hierarchy outperform techniques that incorporate the GO hierarchy. These large classes already contain a good sample of the synonymic terms used to describe the localization, thus using the GO hierarchy does not improve F-measure. In this case extra terms depreciate the value of important terms and creates a poorer quality classifier. I have shown that for each localization class, it is best to consider several ways of representing the text. Large classes will probably perform well with redundancy and no extra processing, whereas smaller classes should use TFIDF and can benefit from the information encoded in the GO hierarchy.

Chapter 6

Conclusion

6.1 Discussion of Results

The field of molecular biology has been moving at an astounding pace. The ability to sequence entire genomes has gone from a pipe-dream to a reality and is now being undertaken in many labs across the globe. In 1995, the first genome was sequenced, that of Haemophilus influenzae [17]. As of November 2006 there are 456 complete genomes published and over 1700 ongoing sequencing projects for the genomes of other organisms [30]. Each one of these genomes represents thousands of proteins waiting to be identified and characterized.

Protein characterization is also becoming more streamlined. Assays have been developed in the last 15 years that allow scientists to more quickly identify a protein's localization within a cell [10]. These techniques have been used in large scale efforts to determine the localization of every protein in a proteome [25]. This sort of high-throughput science was considered science fiction even two decades ago.

When an area of study begins to move quickly, there is a danger that important discoveries may become lost. Finding relevant research can be as difficult as conducting new research. The techniques covered in this dissertation address the classic case of a needle in an ever-growing haystack. It is necessary to develop a way to organize the haystack so that one only has to search through some small subset of the straw in order to find related research. When information is easy to find and access, the seeds of collaboration are sown. Great minds can come together, leverage the power of each other's work and possibly produce results more effectively.

This study combines two areas of Computing Science (machine learning and natural language processing) with a term hierarchy created by biologists. In fact, this thesis is a collaboration of scientific areas in and of itself. Without proper organization of data, this type of scientific fusion would be impossible. To the benefit of this research, the Internet has brought together a vast amount of information. Search engines have assisted in making the information on the Internet more accessible. Ease of retrieval allows for self-education and enhances the ability of experts to collaborate more readily. Text classification for biological journal abstracts is an attempt to reduce the amount of

information one needs to search through in order to find relevant research results. Text classification summarizes an abstract in terms of its strongly indicative words and also allows biologists to filter abstracts based on their classification.

My work has shown that, in certain cases where textual data is limited, the GO hierarchy's ability to aid in synonym resolution and term generalization improves text classification. These techniques can improve predictions of subcellular localization for the proteins discussed in biological journal abstracts. In cases where text is abundant, the GO is not effective. This loss of effectiveness is due to the fact that the journal abstracts already contain many occurrences of synonyms and specific terms. Thus, the necessary terms are prevalent enough in these large bodies of text that a classifier can learn the utility of these words, without the assistance of the GO. I also extended the work of Leopold and Kindermann [29] to the biological domain and showed that the redundancy importance weight measure is often better than TFIDF in text classification of biological journal abstracts when the set of positive data is large.

6.2 Future Work

6.2.1 Incorporating Text Classification into PA's classifiers

Just as MultiLoc combines the predictions of their text classifier with several other classifiers to predict the localization of a protein, PA could use the advancements in text classification presented here to improve its predictors. The predictions of a text classifier could become another feature for PA's SVM, or a super-classifier could be built to mix the predictions of PA and the NLP predictor. The super-classifier could be as simple as a voting mechanism, breaking ties on some measure of confidence in the prediction. Alternatively, the super-classifier itself could use an SVM or some other machine learning algorithm, where the feature vector ($\vec{x}$) contains a representation of the predictions of each of the sub-classifiers. In addition, there is an entire area of machine learning research devoted to combining classifiers, and any of the techniques presented in that body of work might be applied.

6.2.2 Including Other Term Hierarchies

As mentioned in Section 1.3.2, other biological term hierarchies exist (MeSH, enzyme class system), aside from the GO. The utility of each of these hierarchies could be tested, and the usefulness of using several hierarchies together assessed. I have demonstrated that using a term hierarchy can benefit text classification for a technical language like that used in biological journal abstracts, but the research presented here is not restricted to biological writing. Any area that has a lexical resource comparable to the GO could benefit from synonym resolution and term generalization. It would be interesting to see if the techniques developed here generalize to non-technical text classification. For example, WordNet [15] is a lexical database of common English words. WordNet is like the GO

hierarchy, in that it encodes some of the properties that the GO encodes (synonyms, term generalization). The techniques presented here could use WordNet to aid in the classification of non-technical writing, like news articles. Alternatively, WordNet could be leveraged in the biological domain to test if the information encoded in it compliments the technical words in biological term hierarchies.

6.2.3 Using an NER system

The usefulness of a term hierarchy is limited by how up to date it is. Because it is maintained by humans, it is natural that the GO hierarchy's contents will lag slightly behind the evolution of scientific language. An accurate Named Entity Recognition (NER) system could augment the term hierarchy and pick out additional multi-word phrases that are otherwise lost during white-space tokenization. For example, in the case study in Section 5.2.1 contains the phrase "lysosomal acid phosphatase", which is the name of a particular kind of protein known to exist in many eukaryotes including humans, rats and mice. The GO hierarchy will match each of the words in the phrase "lysosomal acid phosphatase", but will fail to identify that the three words together have special meaning. An NER system can pick out phrases like "lysosomal acid phosphatase" which can be used as additional features by a text classifier. While an NER system could not perform synonym resolution or term generalization, the additional features that an NER system identifies might benefit text classification for biological journal abstracts.

6.2.4 Investigating Other Classifications

My research only tested text classification in one specialized area: subcellular localization of animal proteins. An obvious extension is to test the techniques outlined here on subcellular localizations of other organisms. One could also apply these techniques to other classification problems, like protein function or a protein's relation to disease states.

6.3 Summary

Scientific advancements in many areas of research build on each other and expedite the pace of discovery. This dissertation has shown another way in which scientific advancements from different areas (natural language processing, machine learning, biology) can be combined to improve text classification for biological journal abstracts. I used the GO hierarchy in two ways to improve text classification. One, the GO can be used as a thesaurus to perform synonym resolution. Two, the hierarchy's DAG structure can be used to resolve specific terms to broad concepts, as in term generalization. I explored two importance weight measures, TFIDF and redundancy. In general, I have shown that each of TFIDF and redundancy can be used with the information contained in the GO to create more accurate text classifiers. In special cases, where the size of the positive data set is large, I found that it is best to use the redundancy importance weight measure and to forgo adding additional features created using the GO.

In most cases, synonym resolution provides some benefit over baseline. Similarly synonym resolution and term generalization together show an improvement over baseline. Occasionally, term generalization and synonym resolution together show a significant improvement over synonym resolution alone. However, except in the case of very large classes, the F-measure is never significantly worse for classifiers trained on Data Sets that incorporate both synonym resolution and term generalization. Because occasionally the F-measure is significantly better, and never is it significantly worse, it is advantageous to use both synonym resolution and term generalization in practice.

Although this research developed techniques exclusively for biological text, the concepts presented here extend to other areas of text where term hierarchies exist and to other natural language processing applications that might benefit from synonym resolution and term generalization. For example, resources (like WordNet) that exist for non-specialized English words could be used with the techniques presented here to improve general text classification. Successful science is borne of creativity and resourcefulness. This work illustrates how several different resources - Swiss-Prot, biological journal abstracts, the Gene Ontology - can be used to address problems above and beyond those for which they were developed.

Bibliography

[1] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2004.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, October 1990.

[3] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, September 1997.

[4] John Aubrey. *Brief Lives*. History of Economic Thought Books. McMaster University Archive for the History of Economic Thought, 1949. available at http://ideas.repec.org/b/hay/hetboo/aubrey1949.html.

[5] O.T. Avery, C.M. MacLeod, and M. McCarty. Studies on the chemical nature of the substance inducing transformation of pneumococcal types. induction of transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type iii. *Journal of Experimental Medicine*, 79:137–158, 1944.

[6] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D. Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik L. L. Sonnhammer, David J. Studholme, Corin Yeats, and Sean R. Eddy. The Pfam protein families database. *Nucleic Acids Res*, 32 Database issue, January 2004.

[7] LE Baum, T Petrie, G Soules, and N Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[8] Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

[9] M Boguta, LA Hunter, WC Shen, EC Gillma, NC Martin, and AK Hopper. Subcellular locations of mod5 proteins: mapping of sequences sufficient for targeting to mitochondria and demonstration that mitochondrial and nuclear isoforms commingle in the cytosol. *Molecular Cell Biology*, 14(4):2298–2306, April 1994.

[10] M Chalfie, Y Tu, G Euskirchen, WW Ward, and DC Prasher. Green fluorescent protein as a marker for gene expression. *Science*, 263(5148):802–805, February 1994.

[11] N. Collier, H.S. Park, N. Ogata, Y. Tateishi, C. Nobata, T. Ohta, T. Sekimizu, H. Imai, K. Ibushi, and J. Tsujii. The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 271–272, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[12] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[13] Susan T. Dumais, John Platt, David Hecherman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. 7th International Conference on Information and Knowledge Management CIKM*, pages 148–155, 1998.

[14] Richard Durbin. *Biological sequence analysis : probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

[15] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May 1998.

[16] Robert D. Finn, Jaina Mistry, Benjamin Schuster-Böckler, Sam Griffiths-Jones, Volker Hollich, Timo Lassmann, Simon Moxon, Mhairi Marshall, Ajay Khanna, Richard Durbin, Sean R. Eddy, Erik L. L. Sonnhammer, and Alex Bateman. Pfam: clans, web tools and services. *Nucleic Acids Research*, 24:D247–51, January 2006.

[17] RD Fleischmann, MD Adams, O White, RA Clayton, EF Kirkness, AR Kerlavage, CJ Bult, JF Tomb, BA Dougherty, and JM Merrick. Whole-genome random sequencing and assembly of haemophilus influenzae rd. *Science*, 269(5223):469=512, July 1995.

[18] C Geier, K von Figura, and R Pohlmann. Structure of the human lysosomal acid phosphatase gene. *European Journal of Biochemistry*, 183(3):611–616, August 1989.

[19] F Griffith. The significance of pneumococcal types. *Journal of Hygiene*, 1928.

[20] D.G. Higgins, J.D. Thompson, and T.J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting,position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[21] T Hill and P Lewicki. STATISTICS methods and applications. http://www.statsoft.com/textbook/stathome.html, 2006.

[22] Annette Höglund, Torsten Blum, Scott Brady, Pierre Donnes, John San Miguel, Matthew Rocheford, Oliver Kohlbacher, and Hagit Shatkay. Significantly improved prediction of subcellular localization by integrating text and protein sequence data. In *Pacific Symposium on Biocomputing*, pages 16–27, 2006.

[23] Annette Hoglund, Pierre Donnes, Torsten Blum, Hans-Werner Adolph, and Oliver Kohlbacher. MultiLoc: prediction of protein subcellular localization using n-terminal targeting sequences, sequence motifs and amino acid composition. *Bioinformatics*, 22(10):1158–65, May 2006.

[24] Robert Hooke. *Micrographia : or some physiological descriptions of minute bodies made by magnifying glasses with observations and inquiries there upon.* Science Heritage, 1987.

[25] Russell Howson, Won-Ki Huh, Sina Ghaemmaghami, James V. Falvo, Kiowa Bower, Archana Belle, Noah Dephoure, Dennis D. Wykoff, Jonathan S. Weissman, and Erin K. O'Shea. Construction, verification and experimental use of two epitope-tagged collections of budding yeast strains. *Comparative and Functional Genomics*, 6(1-2):2–16, 2005.

[26] N Jarrous, P S Eder, D Wesolowski, and S Altman. Rpp14 and Rpp29, two protein subunits of human ribonuclease P. *RNA*, 5(2):153–157, February 1999.

[27] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, 1998.

[28] Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the Bio-Entity recognition task at JNLPBA. In *Proceedings of the International Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 70–75, 2004.

[29] Edda Leopold and Jorg Kindermann. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46:423–444, 2002.

[30] Konstantinos Liolios, Nektarios Tavernarakis, Philip Hugenholtz, and Nikos C. Kyrpides. The genomes on line database (GOLD) v.2: a monitor of genome projects worldwide. *Nucleic Acids Research*, 34:D332–4, January 2006.

[31] Zhiyong Lu. Predicting protein sub-cellular localization from homologs using machine learning algorithms. Master's thesis, University of Alberta, 2003.

[32] G Nenadic, S Ananiadou, and J McNaught. Enhancing automatic term recognition through term variation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, 2004.

[33] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[34] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[35] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003.

[36] Henry Scheffe. *The Analysis of Variance*. Wiley-IEEE, 1999.

[37] Theodor Schwann. *Microscopical Researches into the Accordance in the Structure and Growth of Animals and Plants*. The Sydenham Society, 1847.

[38] C.J.A. Sigrist, L. Cerutti, N. Hulo, A. Gattiker, L. Falquet, M. Pagni, A. Bairoch, and P. Bucher. Prosite: a documented database using patterns and profiles as motif descriptors. *Brief Bioinformatics*, 3:265–274, 2002.

[39] Gail Sinclair and Bonnie Webber. Classification from full text: A comparison of canonical sections of scientific papers. In *COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004*, pages 69–72, 2004.

[40] Erik L. L. Sonnhammer, Gunnar von Heijne, and Anders Krogh. A hidden Markov model for predicting transmembrane helices in protein sequences. In *Proceedngs Sixth International Conference on Intelligent Systems for Molecular Biology*, pages 175–182, 1998.

[41] B. J. Stapley, Lawrence A. Kelley, and Michael JE Sternberg. Predicting the sub-cellular location of proteins from text using support vector machines. In *Pacific Symposium on Biocomputing*, pages 374–385, 2002.

[42] Swiss-Prot. http://www.expasy.org/sprot/.

[43] Duane Szafron, Paul Lu, Russell Greiner, David S. Wishart, Brett Poulin, Roman Eisner, Zhiyong Lu, John Anvik, Cam Macdonell, Alona Fyshe, and David Meeuwis. Proteome analyst: Custom predictions with explanations in a web-based tool for high-throughput proteome annotations. *Nucleic Acids Research*, 32:W365–W371, 2004.

[44] Hans van Eenennaam, Ger J. M. Pruijn, and Walther J. van Venrooij. hPop4: a new protein subunit of the human RNase MRP and RNase P ribonucleoprotein complexes. *Nucleic Acids Research*, 27(12):2456–2472, June 1999.

[45] Vladimir N Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[46] A Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[47] Robert A. Wallace, Gerald P. Sanders, and Robert J. Ferl. *Biology: The Science of Life*. Harper Collins, 4th edition edition, 2001.

[48] GuoDong Zhou and Jian Su. Exploring deep knowledge resources in biomedical name recognition. In *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 1–7, 2004.