ABSTRACT

A great deal of the genomics information collected throughout the last years is available by accessing online databases such as Medline. This resources are important for biomedical researches in their everyday activities. The volume of published research is expanding at an increasing rate. The Medline 2005 database contains over 12.5 million records and is currently growing with 500,000 citations each year [10]. However there exists few possibilities other than simply querying the data.

A typical problem that biologists face, are extracting relevant information from all this information available. The relatively new research area of bioinformatics, has introduced automatic computer-based systems to assist these tasks. Typically finding gene and protein terms in text and define their annotations (i.e cell functions and roles) and linking them to the Gene Ontology.

In this thesis we propose a new method for searching for gene products and give annotations associating genes with Gene Ontology codes. Many solutions already exists, using different techniques, however few are capable of addressing the whole GO hierarchy. We propose a method for exploring this hierarchy by dividing it into subtrees, trying to find terms that are characteristics for the subtrees involved. By using a feature selection based on $\chi^2$ analysis, finding the words that appear frequently in certain sub trees. These terms are used as a input to a naive Bayes classifier, producing classification values for the go-nodes at one level of the hierarchy at the time.
Preface

This master thesis has been conducted during spring 2005, it is the completion of a 5 years lasting master studies in computer science at the Norwegian University of Science and Technology. It documents the work on designing and implementing a way to automatically search for gene annotations, represented as Gene Ontology codes.

The resulting system implementation and a mysql database of the Gene Ontology is needed to run the system, and is included on an accompanying CD along with installation instructions.

Considerable hours has been spent on learning the SAX parsing interface to Java, data mining techniques and $\chi^2$ analysis (distribution). With very few experience available to build on, my task was rightfully expected to be a challenge and a immense learning endeavour. While stenuos at times and fun at others, it altogether gave me a unique insight into the field of bioinformatics.

Acknowledgments

My expression of gratitude goes to my supervisors; executive professor Heri Ramampiaro, Dr. Astrid Laegreid and civil engineer (master of science) Henrik Tveit for excellent guidance throughout the project.
### Contents

1 Introduction .................................................. 1
   1.1 Problem definition ....................................... 1
   1.2 Thesis outline ........................................... 2

2 Theoretical foundations ................................. 3
   2.1 Molecular Biology ........................................ 3
      2.1.1 Cells, Amino Acids and Proteins ................. 3
      2.1.2 DNA structure and replication .................... 5
      2.1.3 MicroArray .......................................... 6
      2.1.4 Gene Annotation ..................................... 6
      2.1.5 Gene Ontology ...................................... 6
   2.2 Information Retrieval .................................... 7
      2.2.1 Recall and Precision ............................... 8
      2.2.2 Stemming ............................................ 9
   2.3 Data Mining ............................................... 10
      2.3.1 Naive Bayes ........................................ 10
      2.3.2 Finding the class prior probabilities ............ 11
      2.3.3 Estimating the probability of the data given the class 12
      2.3.4 Chi Square and Feature analysis .................. 12
      2.3.5 Chi square used in genetics ....................... 14
      2.3.6 Naive Bayes Text Classification .................. 15
   2.4 XML ....................................................... 16
      2.4.1 SAX ................................................ 17
      2.4.2 SAX and Java ....................................... 18

3 Previous work .............................................. 20

4 Pre-processing ............................................. 21
   4.1 GO and mysql ........................................... 22
   4.2 Perl scripting .......................................... 22
      4.2.1 Downloading stopwords .......................... 22

5 Our Approach ............................................... 23
   5.1 Sub tree classifier ..................................... 23
   5.2 System overview ........................................ 24

6 Implementation ............................................ 28

7 Results ...................................................... 30
List of Algorithms

1 The dynamic classifier ................................................. 25
2 Parsing of input parameters ....................................... 29
3 Heart of the program .................................................. 29

List of Figures

1 the DNA double helix .................................................. 5
2 The GO hierarchy ...................................................... 8
3 Recall and precision ................................................... 9
4 Parsing ved hjelp av SAX ............................................. 19
5 Overview of the steps involved in pre-processing ............ 21
6 Overview of approach ............................................... 23
7 The subtree natur of the go-graph ................................. 24
8 Class diagram .......................................................... 26
9 A typically usage of the system, showing the flow of the information from the user starts the program at the command line to the output of the classifications ............. 27
10 hierarchical precision/recall ......................................... 33

List of Tables

1 Manual annotation process ......................................... 7
2 Weather data .......................................................... 11
3 Observation values ...................................................... 13
4 Expectation ............................................................... 13
5 $\chi^2$ values ............................................................ 14
6 Genotypes after self-crossing generation F1 .................... 15
7 Probability of a word belonging to a class ...................... 16
8 Java -Xmx500m Annotate 0 -g LGALS3 100 ...................... 31
9 Java -Xmx500m Annotate 1 -g ADH1 100 GO:0016614 ........ 32
10 java-Xmx500m Annotate 1 -g LGALS3 100 .................... 79
11 java-Xmx500m Annotate 1 -g ADH1 400 GO:0016491 ........ 80
12 java-Xmx500m Annotate 1 -t 9288754 4000 ................... 81
1 Introduction

The number of bioinformatics resources have grown rapidly since the discipline first formed. Dealing with amounts of complex information, unmanageable for a scientist without sophisticated knowledge about the area. Bioinformatics is evolving to address issues related to such a complex phenomenon as life itself. Through the Human Genome Project[15], started in 1990 the researchers set out for the goal of finding the composition of the human genome. In June 2000 a rough draft of the genome was completed ahead of schedule and in June 2003 the completion of the human DNA sequence was a fact. Traditionally, focus in bioinformatics has been on sequencing analysis, i.e finding the base sequence compositions of the genome. With this information it is possible to perform similarity (homology) analysis to find common ancestry between organisms.

Lately, new technology like the microarray has made it possible for biologists to study thousands of genes at the time. To analyze the results they need to manually annotate the genes themselves, searching for information about the genes in databases like Medline. Also the enormous amount of heterogeneous biological databases has introduced a need for a common vocabulary. The Gene Ontology has been accepted as such, developed to describe processes, functions and cellular components of biological cells. Currently there exists many tools for automatic extraction of gene and protein names in text, using different methods like: machine-learning, rule-based phrase recognition and dictionary-based solutions. However few systems has been implemented for automatically describing a gene by linking it to gene ontology codes.

1.1 Problem definition

In this thesis we will try to develop a system that can automatically download abstracts from Medline based on a gene search query, and find the relevant annotations, represented as nodes in the Gene Ontology tree. We will try to use the whole GO hierarchy in the process, something that has never been done before. Our goal is to find out if this can be carried out and if not, what obstacles must be overcome before this can come to life. We will have to take a look at different data and text mining approaches, before choosing a method that is feasible.
1.2 Thesis outline

This thesis' focus is on data mining techniques in bioinformatics, involving different fields such as biology, information retrieval, and mathematics. For the novice reader to some of these areas, an introduction is given in section 2. An overview of the present state of art and previous work is given in section 3. In section 4 the pre-processing steps that had to be carried out are stated, whereas in section 5 we introduce our own approach. Some implementations details are listed in section 6, and in section 7 we list the results of our proceedings. Finally we include a discussion in section 8, with conclusion and final remarks.
2 Theoretical foundations

In this section we will give an introduction to molecular biology to the interested reader. A theoretical foundations to the concepts of $\chi^2$ feature analysis and naive Bayes classifier will be provided along with some practical usage. A in site into the world of the popular data format XML, and how to use it efficiently dealing with large biological files will be given.

2.1 Molecular Biology

In molecular biology, the scientist studies life at a molecular level. The field overlaps with other areas of biology, particularly genetics and biochemistry. Molecular biology is about understanding the interactions between the various systems of a cell, including the interrelationship of DNA\(^1\) (i.e., the repository of genetic information), RNA\(^2\) and protein synthesis and learning how these interactions are regulated. Since the late 1950s molecular biologists have learned to characterize, isolate, and manipulate the molecular components of cells and organisms. These components include DNA, the repository of genetic information; to actual structural and enzymatic functions as well as a functional and structural part of the translational apparatus; and proteins, the major structural and enzymatic type of molecule in cells. DNA is the major store of genetic information, and is transcribed into RNA which in turn is translated into a protein. This sequence of events is often referred as the central DOGMA of Molecular Biology. In biology the genome is the hereditary information that is encoded in the DNA. This includes both the genes and the none coding regions.

2.1.1 Cells, Amino Acids and Proteins

In 1839 the German biologists Mathias Scheliden and Theodor Schwann[12] integrated the growing body of information on the universal occurrence of cells into one of the first great unifying theories of biology, the cell theory. It stated that all organisms are composed of cells and that all living cells are structurally similar to one another. Some years later

\(^1\)DNA-deoxyribonucleic acid  
\(^2\)RNA-ribonucleic acid
Rudolf Virchow added a third principle when he concluded that all cells arise from the division of preexisting cells. The cells share the following functional characteristics:

1. Cells maintain a selective barrier, called the plasma membrane, which separates the inside of the cell from the external environment.
2. The genetic information is stored in molecules of DNA and is duplicated prior to cell division.
3. Cells contain catalyst called enzymes, which speed up chemical reactions involved in the synthesis and breakdown of organic molecules. These reactions are called metabolism.
4. Cells almost always exhibit some type of mobility. Sometimes this includes movement of the cell as a whole.

In spite of the basic similarities shared by all cells types, one of the biggest difference is the way the genetic material is organized. This has led to the division of the cellular world into eukaryotic and prokaryotic cells. The genetic material of prokaryotic cells is not membrane enclosed and include all forms of bacteria. Eukaryotic cells have the bulk of genetic information in a nucleus surrounded by a double-membrane envelope and consist of single-celled organisms as well as the cells of multicellular plants and animals.

The nucleus of eukaryotic cells has intertwined chromatin fibers, which contain the DNA molecules that store most of the cell's genetic information. During cell division the chromatin fibers and nucleoli condense into compact structures knows as chromosomes. Most of the molecules that make up the living cells have been found to be constructed from only four different chemical building blocks: sugars, fatty acids, nucleotides and amino acids. Cells contain two major classes of nucleic acids refferd to as DNA and RNA. The capital letters A, G, C, T and U are commonly used to refer to adenine, guanine, cytosine, thymine and uracil which are nitrogenous bases employed in the construction of nucleic acids. A, G and C are present in both DNA and RNA, whereas the fourth base is T in DNA and U in RNA. Amino acids are the building blocks of proteins. Twenty different amino acids are used in construction of protein molecules. Amino acids are added to from longer chains called polypeptides. An enormous number of different proteins can be made using the 20 amino acids found in cells. A proteins linear sequence of amino acids is knows as its primary structure. Nucleotide base sequences in DNA code for the amino acid sequences of protein molecules. Once a DNA base sequence has been
determined, the amino acid sequence encoded by that DNA segment can be inferred.

2.1.2 DNA structure and replication

Erwin Chargaff discovered that the amount of adenine and thymine tended to be present in equal amounts, as did guanine and cytosine. This A=T, G=C pattern came to be known as Chargaff's rule. The importance of this structure became apparent later when a three-dimensional model of the structure of DNA was published. It became clear that two intertwined helical chains of DNA could be held together by hydrogen bonds, later referred as the DNA double helix structure. The two DNA chains run in opposite direction, one runs in what is called 5' → 3' direction while the other runs in 3' → 5' direction. The two chains of the DNA double helix are said to be complementary to each other. DNA is replicated by a semi-conservative mechanism based on complementary base pairing. The two strands of the DNA double helix are separated from each other prior to cell division, and each strand then functions as a template that dictates the synthesis of a new complementary DNA strand using the base pairing rules.

![Figure 1: The figure shows the structure of the DNA double helix. The connection between A to T, and G to C can be seen. The figure is retrieved from the U.S National Library of Medicine's web pages.](image-url)
2.1 Molecular Biology

2.1.3 MicroArray

Classical methods in molecular biology only look at one gene at the time. Large scale information such as gene relations cannot be inferred. In the mid 1990s microarray, or DNA chip was developed, making it possible to analyze thousands of genes in one experiment. The gene expression on a whole genome scale can be monitored simultaneously. Microarrays are simply ordered sets of DNA molecules of a known sequence. They can consist of a few hundred to thousands of sets. Each individual feature goes on the array at a precisely defined location. The identity of the DNA molecule fixed to each feature never changes.

2.1.4 Gene Annotation

Annotation is a widely used term normally used to describe metainformation i.e a explanatory note or commentary. Through gene annotations, biologist describe gene roles, using terms from a controlled vocabulary. Pubmed and other databases contain articles describing biological experiments. Knowing that a third of the human genes already have been studied we can save time by using existing research results. The biologist need to read research publications and search online databases to discover the roles of the genes they are working with. This manual annotation process is cumbersome and time demanding, it involves the phases described in table 1.

2.1.5 Gene Ontology

Biologist currently waste a lot of time and effort searching for available information about a small area of research. The wide variations in terminology makes in even harder for computers or people to find relevant information. One database can describe a term as translation whereas another one uses the term synthesis. The Gene Ontology (GO)[5] is a project which goal is to address the need for consistent descriptions of gene products in different databases. The project began as a collaboration between three model organism databases: FlyBase (Drosophila), the Saccharomyces Genome Database (SGD) and the Mouse Genome Database (MGD) in 1998. Since then, the GO Consortium has grown to include many databases, including several of the world's major repositories.
2 THEORETICAL FOUNDATIONS

2.2 Information Retrieval

<table>
<thead>
<tr>
<th>Phase</th>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Information retrieval</td>
<td>Finding and accessing the relevant databases containing the gene function information</td>
</tr>
<tr>
<td>2</td>
<td>Information extraction</td>
<td>Downloading the relevant information from the database sources.</td>
</tr>
<tr>
<td>3</td>
<td>Describing the information through an ontology</td>
<td>Systemising and categorising the functions into a predefined ontology.</td>
</tr>
<tr>
<td>4</td>
<td>Source referencing</td>
<td>Referring to evidence that proves the assigned annotation.</td>
</tr>
<tr>
<td>5</td>
<td>Ranking</td>
<td>Stating a ranking of the annotations' validity.</td>
</tr>
</tbody>
</table>

Table 1: The figure shows the different phases of manual annotations done by biologists for plant, animal and microbial genomes. A gene product can have one or more molecular functions, be used in one or more biological processes and may be associated with one or more cellular components. GO is a directed acyclic graph (DAG) with a hierarchical structure. It is possible for a node to appear in several branches of the hierarchy and have more than one parent node. The structure is shown in figure 2.

2.2 Information Retrieval

Information Retrieval (IR) is defined in[1] as the representation, storage, organization of, and access to information items. The goal of a IR system is to return the relevant information based on a users search, and to make sure that the information is as easy as possible to retrieve. Normally information system are build on index terms to be able to index and retrieve the elements in documents. A index is normally a keyword that gives a lot of information on its own, normally a noun and is used as a occurrence in a table to search for information. Different techniques exists for indexating e.g index files, suffix arrays and signature files. Inverted files[14, 7] has proven to be the best choice many of the time a index structure is used.
2.2 Information Retrieval

2.2.1 Recall and Precision

Many times we want to measure how good a IR system perform.

- Precision is the number of retrieved documents divided by the number of documents which is relevant and is a measure of the usefulness of a hitlist.

\[
Precision = \frac{|Ra|}{|A|} \quad (1)
\]

- Recall is the number of relevant documents divided by the number of documents retrieved and is a measure of the completeness of the hitlist.

\[
Recall = \frac{|Ra|}{|R|} \quad (2)
\]

Recall is 100% when every relevant document is retrieved. In theory, it is easy to achieve good recall: simply return every document in the collection for every query! Therefore, recall by itself is not a good measure of the quality of a specific search. The key in building better retrieval engines
is to increase precision without sacrificing recall. The relation between precision and recall is shown in 3.

High recall is not always needed, since people commonly do not need all relevant items.

![Figure 3: Precision and recall](http://www.tartarus.org/~martin/PorterStemmer/)

### 2.2.2 Stemming

A word can have many variants e.g. plurals, gerund forms and past tense suffixes. Many times it is desirable to remove such suffixes and treat every occurrence of the word as the same. Syntactical variations prevent a perfect match between a query word and a respective document word. The stem of a word is what is left after removal of its affixes (i.e. prefixes and suffixes). Many times stems can improve retrieval performance by reducing variants of the same word to a common concept. There exists however some doubts about the benefits of stemming. Frakes et.al[23] could not find a satisfactory conclusion when comparing eight distinct studies on the potential benefits of stemming. A common version of stemming is known as Porter Stemming. The algorithm was originally described in[13] and is currently available online\(^3\) in many different programming languages. Saetre et al[18] pointed out that in many cases it works poorly when used on protein and biological names, since they are often rooted in Latin or have acronyms as their name or symbols.

---

\(^3\) [http://www.tartarus.org/~martin/PorterStemmer/]
2.3 Data Mining

Data Mining is all about finding structure in data sets and rules that describe them. Text mining is many times looked upon as part of data mining, but dealing with unstructured text only whereas the focus on data mining is on structured data. Text mining allows exploration of unstructured data like publications and reports. Text mining is differentiated from information retrieval in that it examines the relationships between specific kind of information contained both within and between documents. Text mining is closely linked to the field of Natural Language Processing which goals is complete natural language understanding. The nature of human language can be quite complex, introducing a number of difficulties along the way.

In biomedical text the object is often to extract terms like gene and protein names, or interactions among them. A POS tagger is often used to find nouns, verbs and other part of the sentences. Parallelizing the growth of the increase in biomedical literature is the growth in biomedical terminology. Many biomedical entities have multiple names and abbreviations. Some text mining tasks could be done more efficiently if all of the synonyms and abbreviations for an entity could be mapped to a single term representing the concept.

Many classification techniques exists e.g Naive Bayes, k-nearest neighbor and Decision Trees. We will concentrate on the Naive Bayes classifier and use data below shown in table 2.

In this dataset, the attributes are outlook, temperature, humidity and windy and the outcome is whether to play or not. We are interested in classifying the data into two classes, one where the attribute play has the value yes, and the other where it has the value no. The classification will be based on the values of the attributes, and is calculated in the below subsection.

2.3.1 Naive Bayes

Bayes uses all attributes and allow them to make contributions to the decision that are equally important and independent of one another. Bayes’s rule says that if you have a hypothesis $H$, and evidence $E$ then:

$$P[Y|X] = \frac{P[X|Y]P[Y]}{P[X]} \quad (3)$$

10
Table 2: Weather data

<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>no</td>
</tr>
</tbody>
</table>

Where \( P[X] \) denotes the probability of an event \( X \), and \( P[X|Y] \) denotes the probability of class \( i \) conditional on another event \( Y \). In a Bayesian classifier which assigns each data instance to one of \( m \) classes \( C_1, C_2, C_3, \ldots, C_m \), a data instance \( X \) is assigned to the class for which it has the highest posterior probability conditioned on \( X \) i.e \( X \) is assigned to class \( C_i \) if and only if:

\[
P(C_i|X) > P(C_j|X) \quad \text{for all } j \text{ such that } 1 \leq j \leq m, j \neq i. \quad (4)
\]

### 2.3.2 Finding the class prior probabilities

From the frequencies in the data, we can estimate the a priori probabilities.

\[
P(\text{yes}) = P(C_1) = \frac{9}{14} \quad (5)
\]

\[
P(\text{yes}) = P(C_2) = \frac{5}{14} \quad (6)
\]
2.3 Data Mining

2.3.3 Estimating the probability of the data given the class

It can be very computationally expensive to compute the $P(X|C_i)$. If each component of $x_k$ can have one of $r$ values, there are $r^n$ combinations to consider for each of the $m$ classes. In order to simplify the calculation, the assumption of class conditional independence is made, i.e. that for each class, the attributes are assumed to be independent. This assumption allows us to write

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \quad (7)$$

Let us consider the probability of the first data instance in Table 1, given that play = no. We have

$$= P(X = (sunny, hot, high, false)|C_2) = P(x_1 = sunny|C_2) \times P(x_2 = hot|C_2) \times P(x_3 = high|C_2) \times P(x_4 = false|C_2)$$

$$= \frac{3}{5} \times \frac{2}{5} \times \frac{4}{5} \times \frac{2}{5}$$

$$= \frac{48}{625}$$

We can put this together with out known prior probability for class $C_2$ to obtain:

$$P(C_2|X = (sunny, hot, high, false)) = \frac{48}{625} \times \frac{5}{14} = \frac{240}{8750} \quad (8)$$

2.3.4 Chi Square and Feature analysis

Feature selection is all about choosing a limited number of words that are more important than others. Typically stopwords are removed or a POS tagger can be used to find group of words like nouns. However feature selection in biomedicine texts are not a easy subject, many synonyms for gene/protein names exists. Inconsistent expressions occur frequently like: c-jun or c-Jun or c jun or (cJun). Explanatory long expression come in many different shapes:

- the Ras guanine nucleotide exchange factor Sos
- the Ras exchanger Sos

Many times the expressions are mixed with prepositions and conjunctions:

- p85 alpha subunit of PI 3-kinase
The order of words are often changed and abbreviations are used differently:
- heterogeneous ribonucleoprotein K
- hnRNP-K

In general the biological text has a lot of capital and numerical characters. And as pointed out by Setre[18] the process of tokenization (i.e. splitting the text into words) is difficult because of the frequently use of parentheses.

$\chi^2$- is often used in context with feature selection. It is a non-parametric test of statistical significance for bivariate tabular analysis (also known as crossbreaks)[4] Any appropriately performed test of statistical significance lets you know the degree of confidence you can have in accepting or rejecting an hypothesis. Normally, we are trying to find out whether or not two samples are different enough in some aspect of their behavior that we can generalize from our samples that the populations from which our samples are drawn are also different in the behavior or characteristic. The table 5 shows three GO-categories and 4 words that appear a specific number of times in each category, called the observed frequencies.

<table>
<thead>
<tr>
<th></th>
<th>RNA</th>
<th>DNA</th>
<th>helicase</th>
<th>p68</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:1</td>
<td>12</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>GO:2</td>
<td>2</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>GO:3</td>
<td>2</td>
<td>1</td>
<td>9</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>Total</td>
<td>16</td>
<td>14</td>
<td>13</td>
<td>3</td>
<td>48</td>
</tr>
</tbody>
</table>

**Table 3**: Observation values

The expected frequency in each cell is the product of that cell's row total multiplied by that cell's column total, divided by the sum total of all observations. The table 4 shows the calculated expectation values:

<table>
<thead>
<tr>
<th></th>
<th>RNA</th>
<th>DNA</th>
<th>helicase</th>
<th>p68</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:1</td>
<td>5,6</td>
<td>4,9</td>
<td>4,5</td>
<td>1,0</td>
</tr>
<tr>
<td>GO:2</td>
<td>5,9</td>
<td>5,2</td>
<td>4,8</td>
<td>1,1</td>
</tr>
<tr>
<td>GO:3</td>
<td>4,5</td>
<td>3,9</td>
<td>3,7</td>
<td>0,8</td>
</tr>
</tbody>
</table>

**Table 4**: Expectation

The next thing we need to do is measure the size of the difference between the pair of observed and expected frequencies in each cell.
More specifically, we calculate the difference between the observed and expected frequency in each cell, square that difference, and then divide that product by the difference itself. The formula can be expressed as:

$$\left(\frac{(O - E)^2}{E}\right)$$  \tag{9}

Where $O$ is the observed values from table 5 and $E$ is the expectation values from table 4. Squaring the difference ensures a positive number, so that we end up with an absolute value of differences. By using the formula we get the result shown in table 5:

<table>
<thead>
<tr>
<th></th>
<th>RNA</th>
<th>DNA</th>
<th>helicase</th>
<th>p68</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:1</td>
<td>7.4</td>
<td>3.1</td>
<td>1.41</td>
<td>0.02</td>
</tr>
<tr>
<td>GO:2</td>
<td>2.6</td>
<td>9.0</td>
<td>1.6</td>
<td>0.01</td>
</tr>
<tr>
<td>GO:3</td>
<td>1.4</td>
<td>2.2</td>
<td>7.7</td>
<td>0.03</td>
</tr>
<tr>
<td>Total</td>
<td>11.4</td>
<td>14.3</td>
<td>10.7</td>
<td>0.06</td>
</tr>
</tbody>
</table>

Table 5: $\chi^2$ values

As expected the $\chi^2$ value are much lower for the term p68, that had few occurrences and had the same number of occurrences in every class.

### 2.3.5 Chi square used in genetics

The $\chi^2$ analysis is a commonly used method in genetics, and we will include an example of this usage for the interested reader.

Suppose that we want to test the results of a Mendelian genetic cross[6]. We start with 2 parents of genotype AABB and aabb (where A and a represent the dominant and recessive alleles of one gene, and B and b represent the dominant and recessive alleles of another gene).

We know that all the F1 generation (first generation progeny of these parents) will have genotype AaBb and that their phenotype\(^4\) will display both dominant alleles\(^5\).

This F1 generation will produce 4 types of gamete (AB, Ab, aB and ab), and when we self-cross the F1 generation we will end up with a variety of F2 genotypes (see the table below).

---

\(^4\)(the genes that an organism possesses)

\(^5\)One of the different forms of a gene that can exist at a single locus (position)
2 THEORETICAL FOUNDATIONS

2.3 Data Mining

<table>
<thead>
<tr>
<th></th>
<th>AB</th>
<th>Ab</th>
<th>aB</th>
<th>ab</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>AABb</td>
<td>AAbb</td>
<td>AaBB</td>
<td>AaBb</td>
</tr>
<tr>
<td>Ab</td>
<td>AABb</td>
<td>AAbb</td>
<td>AaBB</td>
<td>Aabb</td>
</tr>
<tr>
<td>aB</td>
<td>AaBB</td>
<td>AaBb</td>
<td>aaBB</td>
<td>aaBb</td>
</tr>
<tr>
<td>ab</td>
<td>AaBb</td>
<td>Aabb</td>
<td>aaBb</td>
<td>aabb</td>
</tr>
</tbody>
</table>

Table 6: Genotypes after self-crossing generation F1

All these genotypes fall into 4 phenotypes, shown by colours in the table: double dominant, single dominant A, single dominant B and double recessive. We know that in classical Mendelian genetics the expected ratio of these phenotypes is 9:3:3:1. So we conclude that the hypothesis was correct.

2.3.6 Naive Bayes Text Classification

We are now going to take a look at how we can use Naive Bayes with text classification. The probability of each word appearing in a document of a certain class is estimated directly from the training data. We need to find a probability of a word in a certain class. Following the same methods as in[17] we get:

$$P(w|c) = \frac{s + \sum_{d \in C} I(w, d)}{s + N_{d \in C}} \quad (10)$$

Here $N_{d \in C}$ is the number of documents in the class and $s$ is a constant to avoid a divide by 0. An example of how this might look like is given in table 7.

Further we can easily calculate the probabilities of a document belong to the classes by using a similar formula as in 8. If we take as example a document having all of the terms DNA, RNA and helicase we get:

$$P(GO:1|d) = 3/9 \times P(d|GO:1)$$
$$= P(d|GO:1) = P(RNA|GO:1) \times P(DNA|GO:1) \times P(helicase|GO:1)$$
$$= 3/9 \times 1 \times 0.41 \times 0.8 = 0.108$$
2.4 XML

XML (Extended Markup Language) is a simplified version of SGML (Standard Generalized Markup Language) technology that is often used in applications involved in transferring information. The advantage with XML is that it is easy to understand, process and generate. HTML (Hyper Text Markup Language) is also an instance of SGML, but cannot be expanded with tags like XML. First and foremost HTML is meant to show data using a web browser whereas XML is more about structuring the data.
XML is semistructured with the structure given by the taggs within the document. Sometimes XML documents have a DTD (Document Type Definition) that gives the structure, but this is not a mandatory. On the other hand XML must minimum be well formed and meet the following demands:

- Beginning and end taggs.
- No nested taggs.

Parsing is the process of reading a XML document, checking for validity and perform operations on the data. A parser can be both validating and non-validating. A validating parser must check for consistence and that every element is according to the DTD. This slows the parsing process down.

There are two different ways to perform the parsing of a document:

- Tree based API, by reading the XML document into memory and building a tree based structure. The application can then navigate in this structure called DOM (Document Object Model). The advantage of DOM parsing is that it is easy to make changes to the data and the XML structure. The disadvantage is that the memory usage is unefficient making parsing of large files troublesome.

- A event driven API, based around different parsing events, e.g start of the document, start of a tagg and end of a tagg. The application implements a content handlers for the different events. The most common used API is called SAX.

### 2.4.1 SAX

The parser sends the different events on to a content handler that processes the information. The original document is not changed, but SAX has methods for changing and manipulating the data involved. SAX divides the parsing into different linear event whereas:
2.4 XML

The advantage of this is that large data can be dealt with, and it is not needed to save the data in memory or hard drive. The SAX processing involves:

- Making a content handler
- Making a SAX parser
- Assigning the content handler to the parser.
- Parsing of the document, sending the different events to the content handler.

Because the application doesn’t store the information it is not possible to change it or go back in the data flow.

2.4.2 SAX and Java

A introduction on how to use SAX with Java is given in [8]. The release of Java 1.4 came with a build in parser in the API. The SAX API is divided into two interfaces, XMLReader that represents the parser and ContentHandler that takes input data from the parser. Each time the parser reads a start tagg the startElement() method is called, then when text is read the character() method is called, and finally when a end tagg is read, the endElement method is called. However with by using the helping class DefaultHandler, it is not needed to implement all of the methods, but only the ones needed. Figure 4 shows a overview of the parsing process:

The instantiation of the parse driver can be done in three ways:

- Direct call to the driver (only possible with Java 1.4 or later)
- Specify the driver at start
- Specify the driver as an argument to the method createXMLReader()
The natural choice is to use the build in parser, well documented in [3]. A system for filtering of biological data in XML-format was developed by the author [20], and the reader is referred to this for further details.
3 Previous work

The research seems to be concentrated on two classes of interests. The first is entity identification in text i.e finding biological entities (genes, protein, tissues etc). The second task is more ambitious and focuses on the automatic functional annotation of proteins using the Gene Ontology.

A variety of different approaches has been introduced for automatic computer-based extraction of information to support biomedical researchers. Some use machine learning methods whereas others use dictionary based solutions. Tanabe[21] et al used NLP (Natural language processing) techniques for tagging gene and protein name in full text articles. A rule-based approach requires expert-derived hand written rules, and was used in[16] by Proux et al. Settles[19] used a machine learning system based on conditional random fields (CRFs) with a variety of orthographic and contextual features to find protein and gene terms. This achieved a 78 % recall and a 68 % precision for protein terms. Saetre et al[18] used the search engine Goggle to generate features for protein name extraction. A comprehensive overview of previous work is out of this thesis’ scope. The reader is referred to[9] for a more detailed summary of related publications.

So far the research has been concentrating on finding gene and protein terms and interactions among such. The last years more focusing has been on categorization of genes into the Gene Ontology. In the article “Associating Genes with Gene Ontology Codes Using a Maximum Entropy Analysis of Biomedical Literature”[17] Raychaudhuri et al achieved a document classification of 72.8%, however using only 21 GO nodes at the fourth level of the GO hierarchy. In[11] Kiritchenko et al proposed a hierarchical usage of the Go-tree using a sub tree classifier. No results of their experiments exists.

In many ways we are continuing the research done by Henrik Tveit in his master thesis "Towards an automated procedure for annotation of gene products”[22]. In his work Tveit explored annotation automation through 15 experiments, analyzing both structured and unstructured text. Using a dictionary based approach, searching for co-occurrences of GO definitions in the abstract title and text he achieved a 23 % precision and 10% recall. Using a rule based phraser the results showed 30% recall and 27% precision. Finally using a combination scheme with SVD model he achieved a 38% recall and a 26% precision.
4 Pre-processing

Already given annotation are available from geneontology.com. These are gene products manually annotated to gene ontologies. All the annotations from all the organisms available were downloaded. Only the files that had Pubmed abstracts (i.e. having PMID as evidence code) could be used. With a perl script the annotations that were of type molecular function were extracted, all other information than GO number and pubmed article number were removed, along with duplicates entities and every article that had several GO nodes annotated to it. An article that has several GO-nodes is less useful as evidence, thus introducing a noise factor to the classification process. This left ut with 18126 PMID to GOID annotations.

NCBI - The National Center for Biotechnology Center provides a set of tools that provides access to Entrez data outside a regular web query interface. Two of these are ESearch and EFetch. ESearch can be used to search the databases for e.g. gene names, it returns a set of ids that can be given to EFetch for downloading the information needed. The result can be downloaded in a number of data formats, XML was chosen as the preferred format. The different steps of the pre-processing are summarised in 5.

A small java program (Download.java in the appendix) was made to download these pubmed ids. Because of restrictions only 500 documents could be downloaded at the time.

Figure 5: Overview of the steps involved in pre-processing
4.1 GO and mysql

The complete GO-hierarchy was downloaded from the website of Gene Ontology. At current time there are 17,925 terms, 9,395 biological process, 1,550 cellular component and 6,980 molecular function. Also there are 996 obsolete terms, that we chose not to use. The GO nodes are available in different formats e.g. XML/rdf, flat text file and mysql. After some consideration mysql was chosen as the most suitable format. Also different kinds of databases are available, we chose to download a database containing only the GO terms and its relationships. This is sufficient for our needs. This meant that a mysql server had to be installed and a mysql database was built from the files downloaded (called a mysql dump). Once up and running different queries can retrieve the relationships we need. The queries needed are:

1. Retrieving the children of a node
2. Retrieving the subtree of a node
3. Retrieving the ancestor nodes of a given node

A copy of the mysql database, and installation instructions are included on the CD accompanying this thesis.

4.2 Perl scripting

Perl was chosen to do several small filtering tasks. Especially input and output operations are easily performed with this scripting language. Also the usage of regular expressions make searching for part of strings like “pubid” easy. On the accompanying CD these scripts are included.

4.2.1 Downloading stopwords

A set of stopwords (commonly used English words e.g. is, for and that) were downloaded from pubmed. These words are removed as part of the feature selection process.
5 Our Approach

In this section we will present our ideas and the approach we are going to take in order to do automatic annotation. An overview is given in figure 6. As described in 3 we have downloaded a set of already annotated abstracts from pubmed, dividing them into a training set and a test set. We now want to use this training set to find new annotations, or check if we can find the correct annotations for the test set. Hence our system will be functioning both as an gene annotator and as a document classifier.

![Figure 6: Overview of approach](image)

5.1 Sub tree classifier

We want to take advantage of the structure of the GO-tree and build a dynamic classifier level by level. By using a feature analysis that selects only the words that appear frequently in as few subtrees as possible. The first level is level 0 - the root node, in our case we are only looking at the molecular function part of the tree (i.e molecular function as root node). The first classification we want to do is on level 1, coloured pink in figure 7. We then look at the subtree of the different nodes at level 1. By examining the training set we count the occurrences of the words, and select only those with highest $\chi^2$ value. Finally we do the naive Bayes classification at level 1, using only the words in the document that are also present in the list of $\chi^2$ words. We find probabilities values for our documents (i.e the abstracts downloaded automatically after searching for abstracts on a gene) belonging to the nodes 2.1, 2.2 and 2.3.

As a start we choose to keep the node with highest value and examine its subtree. A more sophisticated approach would be to normalize the
classification values, keeping only the nodes with a value higher than a specific threshold. However this is an easy improvement, and we are more interested in whether or not the methods we are introducing are working. An outline of our method is given in pseudo code in algorithm 1.

5.2 System overview

We will give a short overview of the different processes involved in our system. In the different processes are represented as boxes, and a short description of their methods and tasks will be given below.

- **Annotate** is the main class of the program. It has a method (main) that starts the program. Depending on the parameters given, it initiates ArticleFetch or Search. If a GO-node is not given as a parameter at startup, a cached result of chi words are read from file. And classification is written out on the screen. If a GO-node is given its children nodes are used for classification. Then the Preparse class is called to read in the abstracts and article titles of the complete subtree of this node. To find the children nodes and their subtrees queries are performed involving the JDBC class.
Algorithm 1: General idea, using chi square feature selection on sub trees, building a dynamic classifier level by level, in this scenario we set the number of chi words to be 1000. The algorithm shows the classification at just one level, the user must based on the result, start the classification on the next level.

1: \( d \leftarrow \text{words in document} \)
2: \( \text{chi} \_nr \leftarrow 1000 \)
3: \( \text{for } i = 0 \text{ to number of subtrees do} \)
4: \( \quad \text{HashTable}[i] \leftarrow \text{Words as keys, frequencies as values} \)
5: \( \quad \text{Find } \chi\text{-scores for all the words} \)
6: \( \quad P(w \mid i) \{ \text{Using formula 10} \} \)
7: \( \text{end for} \)
8: \( \text{chi} \_array \leftarrow \text{Select } \text{chi} \_nr \text{ with highest } \chi\text{-scores} \)
9: \( \text{for } i = 0 \text{ to number of subtrees do} \)
10: \( \quad \text{chi} \_hit \leftarrow \text{words that are both in document and in } \text{chi} \_array \)
11: \( \quad \text{print } P(c; \mid d) \{ \text{Using formula 8} \} \)
12: \( \text{end for} \)

- **Search** Searches pubmed for a gene name. Returns a set of ids which is given to the class Fetch.

- **Fetch** downloads a set of pubmed articles that was found in Search. The result is sent through a XML parser and only the article titles and abstracts are extracted and sent to Annotate.

- **ArticleFetch** downloads a specific abstract given by the user. Extracts the abstract text and title, and returns this information to the Annotate class.

- **Preparse** is called from the Annotate class. It get a list of nodes from Annotate and checks if these nodes have abstracts annotated to them (from text file). Using a parser is retrieves all these abstracts’ titles and texts.

- **DefaultHandler** is a superclass that Fetch, Search, ArticleFetch, Preparse and Chi extends. The different parsing methods are accessed through this class. The DefaultHandler class is not shown in figure 8.

- **Jdbc** is the interface to the mysql database. The method getchild returns the child of a given node. The method getchild returns the complete sub tree of a node. And the method getancestor returns all
5.2 System overview

Figure 8: Class diagram

Chi is the class that counts the occurrences of words in a subtree and performs the \( \chi^2 \) square analysis of the data. Only the numbers with the highest values are used, given as a parameter from the user. A classification is finally given on screen to the user.

• *Stem* a implementation of the Porter Stemmer. Some method are added to be able to access it.

In the figure 9 a typical flow of information is shown in a sequence diagram.
Figure 9: A typically usage of the system, showing the flow of the information from the user starts the program at the command line to the output of the classifications.
6 Implementation

The system is implemented using Java 2 Standard Edition 1.4.0 (J2SE). This version has a XML parser included in the API. For the application to run Java 2 Runtime Environment 1.4.0 must be installed. Before the system is run the Java files must be compiled, also a mysql database of the gene ontologys must be installed. J2RE is available from Java Sun's homepages. A mysql database of the gene ontology is available from geneontology's web page and also include on a CD. The system is started with the following command:

```
Java Annotate [0|1] -g [name of gene] [number of chiwords] [GO-node] Java Annotate [0|—1] -t [pubmed article] [number ogof chiwords] [GO-node]
```

The idea is that the program will give a classification of the children of the GO-node given. Then the child with the highest score can be used for further investigation.

The 2 shows a pseudo code of the start of the program after the input parameters is given on the command line. The first parameter is used to activate the stemmer, the second is whether a document classifier or a gene product classifier will be used. The third parameter is a pubid article if used as document classifier otherwise it is a gene name. The fourth parameter is the number of chi words that will be used. If e.g 500 is used the 500 chi words with highest value will be used. The fifth parameter is the level of GO node the classification will start with. The last parameter is optional, if not given the GO node at the highest level will be chosen (i.e GO:0003674). A cached version of chi words will be used to determine the classification values of its children. This case is not covered in 2.

The pseudo code in algorithm in 3 shows a overview of the naive Bayes classification process described in the theoretical foundation section.
Algorithm 2: Parsing of input parameters

1: if 1.option is true then
2:    stem $\leftarrow 1$
3: end if
4: if option nr 2 equals -g then
5:    gene_name $\leftarrow 2$.option
6:    nr_of_chi $\leftarrow 3$.option
7:    go $\leftarrow 4$.option
8:    abstracts $\leftarrow \text{Search}.\text{get_text}(\text{gene_name})$
9:    Annotate(abstracts, go, stem, nr_of_chi)
10: end if
11: if option nr 2 equals -t then
12:    pub_id $\leftarrow 2$.option
13:    nr_of_chi $\leftarrow 3$.option
14:    go $\leftarrow 4$.option
15:    abstract $\leftarrow \text{ArticleFetch}.\text{get_text}(\text{pub_id})$
16:    Annotate(abstract, go, stem, nr_of_chi)
17: end if

Algorithm 3: Heart of the program

1: Annotate(abstracts, go, stem, nr_of_chi)
2: tokenstem(abstracts)
3: words_abstracts $\leftarrow$ construct hash table of words in abstracts
4: for $i = 0$ to number_of_classes do
5:    prob_word_in_class $\leftarrow 1$
6:    for $j = 0$ to $\chi$-words.length do
7:        if words_abstracts contains $\chi$-words $j$ then
8:            number $\leftarrow$ class $i$, $\chi$ - word $j$
9:            number $\leftarrow (1+$number$) \times 1000$
10:           prob_word_in_class $\leftarrow$ prob_word_in_class $\times$ number
11:        end if
12:    end for
13:    classify $\leftarrow$ classify $\times$ number of documents in $i$
14: end for
7 Results

The system was tested on a Linux-based machine, but since Java is platform independent, there should be no problems using it on Windows as well. The configuration of the computer used was as following:

- Operation system: Debian Linux Sid
- Hard disk: 40 GB
- CPU: 2.4 GH
- Memory: 512 MB RAM

7.1 Test set

The system was tested both as a document classifier and as a gene classifier. To test the document classifier, a set of 50 annotated abstracts where used. To test the gene classifier a set of manually annotated genes were used (appendix B).

7.2 Gene classifier

First we tested our system on a level 1 classification, i.e finding the class probabilities of the children of the root node (i.e GO:0003674, molecular function). When a node is not given as the last parameter an already cached version of chi numbers is used, enabling a fast classification. The output is given in Abel 8. As can be seen in the parameter values given at startup we are searching for annotations for the gene LGALS3. In the appendix(B) the manual annotations for this gene is GO:0019863, GO:0016936, GO:0008248 and GO:0005529. All of which have ancestor GO:0005488 (binding) at level 1. We use 100 \( \chi^2 \) numbers with the highest values (i.e words that show strong belongings to one or a few subtrees).

The correct path of e.g GO:0019863 is:

- GO:0003674: molecular function
- GO:0005488: binding
  - GO:0005515: protein binding
  - GO:0019865: immunoglobulin binding
GO:0019863: IgE binding

The correct path in the GO-tree can easily be found, by querying the mysql database or by using tools like Amigo. The table 8 gives a value of 1.2E11 for the correct GO:0005488, but even a higher value for GO:0003824.

<table>
<thead>
<tr>
<th>GO-id</th>
<th>Classification probability</th>
<th>Documents in subtree</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0003774</td>
<td>3547.2004615384617</td>
<td>97</td>
</tr>
<tr>
<td>GO:0003824</td>
<td>3.956548875048E11</td>
<td>10062</td>
</tr>
<tr>
<td>GO:0004871</td>
<td>2.0951941551542308E9</td>
<td>2713</td>
</tr>
<tr>
<td>GO:0005198</td>
<td>1545853.0855384616</td>
<td>447</td>
</tr>
<tr>
<td>GO:0005215</td>
<td>4.65313625784E9</td>
<td>3312</td>
</tr>
<tr>
<td>GO:0005488</td>
<td>1.2169443244746277E11</td>
<td>7491</td>
</tr>
<tr>
<td>GO:0005554</td>
<td>10568.315076923076</td>
<td>128</td>
</tr>
<tr>
<td>GO:0016209</td>
<td>1200.7211538461538</td>
<td>74</td>
</tr>
<tr>
<td>GO:0030188</td>
<td>0.24923076923076923</td>
<td>8</td>
</tr>
<tr>
<td>GO:0030234</td>
<td>2.002978433076923E7</td>
<td>847</td>
</tr>
<tr>
<td>GO:0030528</td>
<td>2.1716950936153847E8</td>
<td>1539</td>
</tr>
<tr>
<td>GO:0030533</td>
<td>0.24923076923076923</td>
<td>8</td>
</tr>
<tr>
<td>GO:0045182</td>
<td>5838.258461538461</td>
<td>110</td>
</tr>
<tr>
<td>GO:0045735</td>
<td>0.002769230769230769</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 8: Java -Xmx500m Annotate 0 -g LGALS3 100

The next test was to use the same input but this time using a stemmer. By setting the first parameter to 1 we tried running with "Java Annotate 1 -g LGALS3 100". The result can be seen in appendix C table 10. The result shows lower classification values, i.e hitting on fewer χ words, but also here the GO category GO:0003834 gets higher value than the correct one.

We now tried doing a classification on level 2 with command Java -Xmx500m Annotate 1 -g ADH1 400 GO:0016491. The result is shown in appendixC table 11. The results are once again not what we had hoped for. It seems like the categories with the highest training abstracts gets the highest classification.
7.3 Document classifier

Next we wanted to try out annotation systems and see if it was suited as a document classifier. If this task gives good result, than the precision and recalls values are easier to calculate. We tried with a number of different settings, the result of using Java -Xmx500m Annotate 1 -g ADH1 100 GO:0016614 is shown in table 9. The output was not what we had hoped for here either.

<table>
<thead>
<tr>
<th>GO-id</th>
<th>Classification probability</th>
<th>Documents in subtree</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0004457</td>
<td>9.273996165155403E75</td>
<td>18</td>
</tr>
<tr>
<td>GO:0016615</td>
<td>1.4105549517062977E72</td>
<td>24</td>
</tr>
<tr>
<td>GO:0016616</td>
<td>4.5902665871035206E119</td>
<td>182</td>
</tr>
<tr>
<td>GO:0016898</td>
<td>2.425728858640503E46</td>
<td>8</td>
</tr>
<tr>
<td>GO:0016899</td>
<td>4.457844881615237E42</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 9: Java -Xmx500m Annotate 1 -g ADH1 100 GO:0016614

7.4 A new method for measuring Precision and Recall

Even though our result were not as we had hoped for, we will give a short description of how we would have measured the precision and recall. It is clear that the normal method for finding precision and recall would produce poor results, not taking into account that a classification to a sibling or a parent of the respective node. In ?? Kiritchenko et al propose a new hierarchical evaluation measure based on two principles:

1. Give higher evaluation for correctly classifying one level down as oppose to staying at the parent node.

2. Give lower evaluation for incorrectly classifying one level down comparing to staying at the parent node.

They propose a hierarchical precision and recall values given as:

\[
HP_i = \frac{TP_i}{TP_i + FP_i} \quad hR_i = \frac{TP_i}{TP_i + FN_i}
\]

(11)

Where HP denotes hierarchical precision and hR denotes hierarchical recall. In figure 10 to the left a parent and a child were classified correctly thus we get perfect precision and recall. In the middle picture an instance
7 RESULTS 7.4 A new method for measuring Precision and Recall

Figure 10: The ellipse at the bottom left represent the real category of a test instance, whereas the arrow is pointing to the category found by our classifier. The path from the root to the correct category is shown in bold. Numbers in the nodes represent the number of instances correctly assigned to the node by the classifier (TP), total assigned by the classifier (TP + FP), and total assigned by the experts (TP + FN). Hierarchical precision hP and hierarchical recall hR is shown below the figure.

is misclassified into a sibling of the correct category, and we get 1/2 for precision and recall. In the right most picture, an instance is misclassified into the parent of the correct category. However since this is the parent of the correct category we get perfect precision, but since only one of the two correct categories is assigned we get half recall.

7.4.1 Precision/recall trade-off

With the proposed hierarchical measure of precision and recall it is possible to find a trade-off depending on the granularity of our search. To get a good precision we can set a high threshold value, stopping early in the graph. To get high recall we push examples to lower levels.
8 Discussion

In this section we will shortly discuss the results that we got.

8.1 Feature selection is complicated

Words counts with the $\chi^2$ feature selection does not give good enough results. If a $\chi^2$ word with high value is not biologically correct it effects the results drastically. We decided to keep all numbers as possible $\chi^2$ words, and did not have time to perform experiments without them, e.g. numbers like 3 and 5 can have biological interest, most numbers do not. The use of a stemmer proved to have little impact on the results.

8.2 Scalability

A data mining classifier can only be used when the number of categories are few. The GO-tree consists of thousands of nodes and Medline has over 12 million records. Most algorithms are not suited for this kind of number. However our divide-and-conquer strategy gets rid of these shortcomings.

8.3 Naive Bayes

Feature selection effectiveness is related to the text categorization algorithm and the dataset. The results conclude that naive Bayes perform weakly on rare categories. It may not be suited for usage with such a noisy feature set.

8.4 Incomplete training data

The training data used, only covers half of the nodes in the molecular function part of the GO-tree. Some nodes have many annotations to abstracts, and some have few. The subtrees at level 1 for GO:0003824 and GO:0005544 have the majority of all annotations. The effect of this is that too many of the chi words are chosen from these parts of the tree. The manually annotations are growing, hence the training set is getting
better and more complete. With this in mind there are possibilities that our system will become better in the future.

8.5 Suggestions for further research

At the time being the system is only available through command line usage. A GUI (graphical user interface) was not developed. This would make the system more accessible for biologists. The ultimate goal was to make it accessible by web, by using JSP (Java server pages). The system can easily be expanded to also include biological processes and cellular components. The system can also be expanded to use full text instead of only abstracts. It has been shown that only a few of the protein interactions contained in the Dictionary of Interacting Proteins (DIP) could be found in Medline sentences [2].

8.6 Conclusion

The system developed is up and running, the time spent to implement it was such that not much testing could be done. Many ideas was not tested, e.g. giving higher weighting to the abstract title compared to the abstract text. We did not have time to investigate other classification methods than naive Bayes, and suspect that we might have gotten better results by other methods. The feature selection we chose by using $\chi^2$ selection must be combined with other methods for better results.

The experiments described in the Results section, did not give as good results as we had anticipated. In ?? Raychaudhuri et al achieved a document classification of 72.8%, however using only 21 GO nodes at the fourth level of the GO hierarchy. Our experiments did not have this kind of limitation. In ?? Kritchenko et al proposed a similar hierarchical usage of the GO-tree using a similar sub classifier that we have implemented. No results of their experiments exists, and we suspect they would get the same results as we have.

Having said that, we still feel that the ideas presented can be useful, and combined with other tools for protein/gene tagging like ?? we expect better results.
References


A appendix

A.1 Java Code

```java
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.io.*;
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import java.util.ArrayList;

public class Fetch extends DefaultHandler{
    public String aen, ben;
    public String tekst = "";
    public int antall = 0;
    public int flagg = 0;

    public Fetch(String a, String b, String go, boolean stem, int chi){
        aen = a;
        ben = b;
        System.out.println("er her");
        try{
            URL url;
            URLConnection urlConn;
            DataOutputStream printout;
            DataInputStream input;
            String urlen = "http://envis.eutelsat.aero/envis/efetch.fcgi?db =pubmed" +"&" + "retmode=xm1" + "&" + "WebEnv=" + aen + "&" + "query_key=" + ben;
            url = new URL (urlen);
            urlConn = url.openConnection();
            urlConn.setDoInput (true);
            urlConn.setDoOutput (true);
            urlConn.setUseCaches (false);
            urlConn.setRequestProperty
            ("Content-Type", "application/x-www-form-urlencoded");
            input = new DataInputStream (urlConn.getInputStream ());
            SAXParserFactory factory = SAXParserFactory.newInstance();
            SAXParser saxParser = factory.newSAXParser();
            saxParser.parse(input, this);
            input.close();
            antall = antall / 2;
            Annotate hoved = new Annotate(tekst, antall, go, stem, chi);
        } catch (SAXException t) {
            System.out.println("Something wrong when parsing");
            System.exit(0);
        } catch (IOException g) {
            System.out.println("IO-fault reading when parsing");
        }
    }
}
```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.util.*;
import java.util.Calendar;
import java.text.DateFormat;
import java.net.*;
import java.io.*;
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import java.util.ArrayList;

public class ArticleFetch extends DefaultHandler{
    public String aen, ben;
    FileOutputStream out; // declare a file output object
    PrintStream p; // declare a print stream object
String pubmedid;
String a,b;
public String pubid;
String tekst = "";
int tall, flagg;
int retestar = 0;
int remax = 500;
int jada:
String alfa;
FileInputStream fstream;
DataInputStream in;

public ArticleFetch (String pubid)
{
    System.out.println("er i test");
    this.pubid = pubid;
    try{
        URL url;
        URLConnection urlConn;
        DataOutputStream printout;
        DataInputStream input;

        url = new URL (urlen);
        urlConn = url.openConnection();
        urlConn.setDoInput (true);
        urlConn.setDoOutput (true);
        urlConn.setUseCaches (false);
        urlConn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
        input = new DataInputStream (urlConn.getInputStream ());

        SAXParserFactory factory = SAXParserFactory.newInstance();

        SAXParser saxParser = factory.newSAXParser();
        saxParser.parse(input, this);
        input.close();
        // Connect print stream to the output stream

    } catch (SAXException t) {
        System.out.println("Something wrong in config file");
        System.exit(0);
    } catch (IOException g) {
        System.out.println("IO-fault reading from config file, make sure it exists and can be read");
        System.exit(0);
    } catch (ParserConfigurationException abb) {
        System.out.println("Something went wrong setting up the parser, make sure you have java JDK1.4 or newer");
        System.exit(0);
    }
}

public String get_article()
{ return tekst;
A APPENDIX A.1 Java Code

```java
public void endDocument() throws SAXException {
    }

public void characters(char[] ch, int start, int length) throws SAXException {
    if (flagg == 0) {
        System.out.println("er her ja");
        tekst = tekst + " " + new String(ch, start, length);
    }
}

public void endElement(String namespaceURI, String localName, String qName) throws SAXException {
    flagg = 0;
}

public void startElement(String namespaceURI, String sName, // simple name (localName)
String qName, // qualified name
Attributes attrs) throws SAXException {
    if (qName.equals("ArticleTitle") || qName.equals("AbstractText")) {
        flagg = 1;
    }
}

import java.awt::*;
import java.applet.*;
import java.io::*;
import java.util.*;
import java.io.*;
import org.xml.sax.*;
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import java.util.ArrayList;

public class Search extends DefaultHandler {
    public int question = 0;
    public int antall;
    public String query_key, webevn, search_goid, tekst;
}
```

Search.java

41
public Search(String searchfor) {
    search = searchfor;
    System.out.println("er i ask");
    try {
        URL url;
        URLConnection urlConn;
        DataOutputStream printout;
        DataInputStream input;
        urlConn = url.openConnection();
        urlConn.setDoInput (true);
        urlConn.setDoOutput (true);
        urlConn.setUseCaches (false);
        urlConn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
        input = new DataInputStream (urlConn.getInputStream ());
        SAXParserFactory factory = SAXParserFactory.newInstance();
        SAXParser saxParser = factory.newSAXParser();
        saxParser.parse(input, this);
        input.close();
    } catch (SAXException t) {
        System.out.println("Something wrong in config file");
        System.exit(0);
    } catch (IOException g) {
        System.out.println("IO-fault reading from config file, make sure it exists and can be read");
        System.exit(0);
    } catch (ParserConfigurationException a) {
        System.out.println("Something went wrong setting up the parser, make sure you have java JDK1.4 or newer");
        System.exit(0);
    }

    Fetch get = new Fetch(webenv, query_key);
    tekst = get.get_text();
    antall = get.get_count();
}

public String get_text(){
    return tekst;
}

public int get_count(){
    return antall;
}

public void characters(char[] ch, int start, int length) throws SAXException {
    if (question==1){

}}
A.1 Java Code

String nystreng = new String(ch.start.length);
webenv = nystreng;
System.out.println(nystreng);
if (question == 2) {
    String nystreng = new String(ch.start.length);
    query_key = nystreng;
    System.out.println(nystreng);
}
public void endElement(String namespaceURI, String localName,
                      String qName) throws SAXException {
    question = 0;
}
public void startElement(String namespaceURI, String sName, // simple name (localName)
                         String qName, // qualified name
                         Attributes attrs)
    throws SAXException {
    // System.out.println(qName);
    if (qName.equals("WebEnv")) {
        question = 1;
    }
    if (qName.equals("QueryKey")) {
        System.out.println("jada");
        question = 2;
    }
} // End of class Happy.

import java.sql.*;
import java.util.ArrayList;

public class Jdbc{
    public ArrayList array;
    public Statement stmt;
    public Connection con;
    public String buffer = "";
    public int tt = 0;
    public Jdbc(){
        connect_database();
    }

    public void connect_database(){
        try {
            Class.forName("com.mysql.jdbc.Driver");
            String url = "jdbc:mysql://localhost:3306/sgwrd";
            con = DriverManager.getConnection(url, "root", "");
            stmt = con.createStatement();
        } catch (Exception e) {
        }
}
public ArrayList get_child(String go){
  try{
    ResultSet rs = stmt.executeQuery("SELECT rchild.* FROM term AS rchild, term AS ancestor, graph_path WHERE graph_path.term2_id = rchild.id and graph_path.term1_id = ancestor.id and ancestor.acc = 'GO:0003674';
    
    ResultSet rs = stmt.executeQuery("SELECT child.* FROM term AS parent, term2term AS term child WHERE parent.acc = " + '"' + go + '"' + "AND parent.id = term2term.term1_id AND child.id = term2term.term2_id");
    array = new ArrayList();
    while(rs.next()){
      String hallo = rs.getString(4);
      array.add(hallo);
    }
    rs.close();
  }
  catch( Exception e ) {
    e.printStackTrace();
  }
  } //end catch
  return array;
} //end main

public ArrayList getter2(String go){
  try{
    ResultSet rs = stmt.executeQuery("SELECT p.* FROM graph_path INNER JOIN term AS t ON (t.id = graph_path.term2_id)INNER JOIN term AS p ON (p.id = graph_path.term1_id) WHERE t.acc = " + '"' + go + '"' + "");
    array = new ArrayList();
    while(rs.next()){
      String hallo = rs.getString(4);
      array.add(hallo);
    }
    rs.close();
  }
  catch( Exception e ) {
    e.printStackTrace();
  }
  } //end catch
  return array;
} //end main

*/
public void test(String node) {
    try {
        ResultSet rs = stmt.executeQuery("SELECT * FROM term WHERE acc = '" + node + "'");
        while (rs.next()) {
            String hallo = rs.getString(3);
            String jada = rs.getString(4);
            if (hallo.equals("cellular_component")) {
                System.out.println(jada + " tam tam " + hallo);
                System.exit(-1);
            }
        }
        rs.close();
    } catch (Exception e) {
        e.printStackTrace();
    } // end catch
}

public ArrayList ancestor(String go) {
    try {
        ResultSet rs = stmt.executeQuery("SELECT p.* FROM graph_path INNER JOIN term AS t ON (t.id = graph_path.term2_id) INNER JOIN term AS p ON (p.id = graph_path.term1_id) WHERE t.acc = '" + go + ")");
        array = new ArrayList();
        while (rs.next()) {
            String hallo = rs.getString(4);
            array.add(hallo);
        }
        rs.close();
    } catch (Exception e) {
        e.printStackTrace();
    } // end catch
    return array;
} // end main

public ArrayList get_rchild(String go) {
    try {
        ResultSet rs = stmt.executeQuery("SELECT rchild.* FROM term AS rchild, term AS ancestor, graph_path WHERE graph_path.term2_id = rchild.id and graph_path.term1_id = ancestor.id and ancestor.acc = '" + go + ") and graph_path.distance > 0"; ");
        array = new ArrayList();
    }
while(rs.next()){
    String hallo = rs.getString(4);
    array.add(hallo);
}
rs.close();
//con.close();
} catch(Exception e) {
    e.printStackTrace();
} //end catch
return array;
} //end main
} //end class Jdbc

public class Chi extends DefaultHandler{
    public int question = 0;
    public String nystreng,h,aaa,a,beta,ceta,ii,gg,qqq;
    public String tidligere = "";
    public BufferedWriter out;
    public HashMap hashMap,hoho,haaa,hiha;
    public int jalla2 = 0;
    public int jalla ,question2;
    public int ja=0;
    public int hh = 0;
    public StringBuffer buffer;
    public int u = 0;
    public int i = 0;
    public int flagg = 0;
    public int[] la ,test ,antallet;
    public String[] oo ,arr;
    public HashMap[] hasj,hasj2,chihasj;
    public HashMap a1,a2,a3;
    public int jaba,jo,tail;
A. APPENDIX A.1 Java Code

```java
public int hi = 0;
pUBLIC double total = 0.0;
public String str, s1, s2, nystreng_old;
public Pattern p;
public String[] chiwords;
public Matcher m;
public int size, totalen;
public double g;
public int ass = 0;
public FileInputStream fstream, fstream2, fStreamen;
public TreeMap treeMap;
public HashMap t;
public BufferedReader ina;
public DataInputStream inen, ina;
public Collection[] colla;
public Weka weka;
public HashMap trening, test2, midler;
public Stem stem;
public Jdbc jdb;
public int chi;
public int index = 0;
public boolean stemmen;
public ArrayList hasArray, rowsum, fifi, fofo, pubider, goer, klasser;
public Chi(ArrayList klasse, Stem stemer, HashMap tren, HashMap test2, 
ArrayList pubide, ArrayList goe, Jdbc jd, boolean stemm, int chinn)
   chi = chinn;
   klasser = klasse;
   stemmen = stemn;
   jdbc = jd;
   trening = trene;
   this.test2 = test2;
   pubider = pubide;
   goer = goe;
   stem = stemer;
   size = klasse.size();
   midler = new HashMap();
   try
   {
      antallet = new int[klasse.size()];
      hasArray = new ArrayList();
      fifi = new ArrayList();
      hashMap = new HashMap();
      treeMap = new TreeMap(hashMap);
      rowsum = new ArrayList();
      hasj = new HashMap[size + 2];
      test = new int[2];
      for(int a = 0; a < size + 2 ; a++)
      {
         hasj[a] = new HashMap(trenning.size());
      }
      Set setting2 = trening.keySet();
      Iterator iter2 = setting2.iterator();
   }
```

47
while (iter2.hasNext()) {
    Object o = iter2.next();
    s4 = o.toString();

    for (int a = 0; a < size; a++) {
        hasj[a].put(s4, "0");
    }

    for (int i = 0; i < size + 2; i++) {
        HashMap hasjsj = (HashMap) hasjArray.get(i);
        hasjsj.put(s4, "0");
        hasjArray.set(i, hasjsj);
    }

    SAXParserFactory factory = SAXParserFactory.newInstance();

    DateFormat df = DateFormat.getTimeInstance();
    int tid = df.getCalendar().get(Calendar.HOUR_OF_DAY);

    int tid2 = df.getCalendar().get(Calendar.MINUTE);
    int tid3 = df.getCalendar().get(Calendar.SECOND);
    System.out.println("START" + " HOUR:" + tid + " MINUTE:" + tid2 + " SECOND:" + tid3);

    SAXParser saxParser = factory.newSAXParser();
    saxParser.parse(new File("ja.txt").this);

    try {
        saxParser.parse(new File("ja.txt").this);
    } catch (SAXException t) {
        System.out.println("Something wrong in config file" + t.toString());
    } catch (IOException g) {
        System.out.println("IO-fault reading from config file, make sure it exists and can be read");
        System.exit(0);
    } catch (ParserConfigurationException a) {
        System.out.println("Something went wrong setting up the parser, make sure you have Java JDK 1.4 or newer");
        System.exit(0);
    }

    System.out.println("Calculating chi words");
    calculate_chi_words();
    classify();
}
public void characters(char[] ch, int start, int length) throws SAXException {
    if (question==3) {
        jaba = Integer.parseInt(new String(ch, start, length));
    }

    for(int a = 0; a<pubider.size(); a++) {
        question2=0;
        question=0;
        int dd = ((Integer)pubider.get(a)).intValue();
        if (jaba==dd) {
            question2 = 1;
            question=0;
            break;
        } else {
            question2 = 0;
            question = 0;
        }
    }

    if (question==2 && question2==1) {
        buffer = new StringBuffer();
        nystreng = String.valueOf(ch, start, length);
        buffer.append(nystreng);
        buffer.append(" ");
        nystreng="";
    }

    if (question==1 && question2==1) {
        //System.out.println(jaba);
        buffer.append(String.valueOf(ch, start, length));
        flagg =1;
        nystreng = "";
    }
}

public void endElement(String namespaceURI, String localName, String qName) throws SAXException {
    if (flagg ==1) {

a = (String) goer.get(index);
index++;
nystreng = buffer.toString();
nystreng_old = nystreng;
    Pattern p = Pattern.compile("(.*\s)(.*$)\s(.*\")");
Matcher m = p.matcher(a);
    if (m.matches()){
        s1 = m.group(1);
        s2 = m.group(2);
    }

    jalla = Integer.parseInt(s1);
    ceta = s2;
    // System.out.println("jalla er " + jalla + "jalla2 er " +
    jalla2);
    
    ArrayList akk = jdbc.ancestor(s2);
    // ArrayList akk = new ArrayList();
    // counter(nystreng,ceta);
    
        for (int i = 0; i < akk.size(); i++){
            for (int a = 0; a < klasser.size(); a++){
                String ai = (String) aklasse.get(i);
                String oi = (String) klasser.get(a);
                if (ai.equals(oi)){
                    // System.out.println(nystreng);
                    counter(nystreng,(String) aklasse.get(i));
                }
            }
            buffer = null;
        }

        flagg = 0;
        question = 0;
    }

    public void startDocument() throws SAXException{
    
    } public void endDocument() throws SAXException{
    DateFormat df = DateFormat.getTimeInstance();
    int tid = df.getCalendar().get(Calendar.HOUR_OF_DAY);
}
```java
int tid2 = df.getCalendar().get(Calendar.MINUTE);
int tid3 = df.getCalendar().get(Calendar.SECOND);
System.out.println("END:" + "HOUR:" + tid + "MINUTE:" + tid2 + "SECOND:" + tid3);
String q = "";
String s = "";
System.out.println(size);

public String[] getwords(){
    System.out.println("length er an " + chiwords.length);
    return chiwords;
}

public ArrayList getklasse(){
    return hasJArray;
}

public ArrayList getklasse(){
    return klasser;
}

public void startElement(String namespaceURI,
String sName, // simple name (localName)
String qName, // qualified name
Attributes attrs)
throws SAXException
{
    if(qName.equals("PMID")){
        question = 3;
    }
    if(qName.equals("AbstractTest")){
        question = 1;
    }
    if(qName.equals("ArticleTitle")){
        question=2;
    }
}

public void counter(String streng,String alla){
    //System.out.println(streng + " " + alla + "%%" + hasj[0].size());
tall = 0;
    for(int g = 0; g < klasser.size(); g++){
        if(((String)klasse.get(g)).equals(alla))
        {tall = g;
        break;
        }
    }
```
A.1 Java Code

```java
// System.out.println("tallet er " + tall);

p = Pattern.compile("\p{Punct}\"\"\")
    m = p.matcher(strength);
    String juha = m.replaceAll("\"\"\"\"");

Pattern splitter = Pattern.compile("\[\]+\"\"\")
    \// Pattern splitter = Pattern.compile("[\+:\];:\\]+\"");
    String[] words = splitter.split(juha);

if(stemmen){
    for(int i = 0; i < words.length; i++){
        words[i] = stemhei(words[i]);
    }
}
else{
    for(int i = 0; i < words.length; i++){
        words[i] = words[i].toLowerCase();
    }
}

for(int y = 0; y < words.length; y++){
    if(has[j].containsKey(words[y])){
        midler.put(words[y], "0");
        // System.out.println("GGGGGGGGGGGGGGGGGGG");
    }
}

Set setting1 = midler.keySet();
Iterator iter1 = setting1.iterator();

while(iter1.hasNext()){
    Object o = iter1.next();
    String s2 = o.toString();
    Object oo = has[j].get(s2);
    String s3 = oo.toString();
    int c = Integer.parseInt(s3);
    c++;
    h = Integer.toString(c);
    has[j].put(s2, h);
}

midler.clear();
    int a = 0;
```
for (int bb = 0; bb < size; bb++)
    if (alla.equals(((String) klasser.get(bb)))
        antall[bb] = antall[bb] + 1;
        hoho = (HashMap) hasjArray.get(bb);
        a = bb;

for (int i = 0; i < words.length; i++)
    if (hoho.containsKey(words[i]))
        try {
            Object ai = hoho.get(words[i]);
            String b = ai.toString();
            int c = Integer.parseInt(b);
            // System.out.println(c);
            c++;
            h = Integer.toString(c);
        }
        catch (Exception e) {
            hoho.put(words[i], h);
        }
    // NB kan ævre feil
    hasjArray.set(a, hoho);

    public void calculate_chi_words()
    {
        int halla = 0;

        for (int i = 0; i < antall.length; i++)
            System.out.println(antall[i]);
    }

    for (int i = 0; i < size; i++)
        System.out.println("#" + size);
    double jo = 0.0;
    Collection collas = ((HashMap) hasjArray.get(i)).values();
    Iterator it = collas.iterator();
    while (it.hasNext())
    {
        Object o = it.next();
        String ss = o.toString();
        double c = Double.parseDouble(ss);
        jo = jo + c;
} 
    rowsum.add(i.,new Double(jo));
    System.out.println("********###############");
    System.out.println("rowsum" + jo);
} 

System.out.println("\ø\ strelsen av klassen" + klasses.size());
ArrayList aiu = new ArrayList();
for(int b = 0; b < klasses.size(); b++)
    if(((Double) rowsum.get(b)).doubleValue() == 0.0)
        System.out.println("rowsum er 0");
    aiu.add(new Integer(b));
}
for(int d = 0; d < aiu.size(); d++)
    integer bbb = (Integer) aiu.get(d);
System.out.println("rowsum er 3");
int aab = bbb.intValue();
klasses.remove(aab-ass);
hasArray.remove(aab-ass);
System.out.println("rowsum er 5");
rowsum.remove(aab-ass);
size--;
ass++;
}
for(int g = 0; g < size; g++)
    total = total + ((Double) rowsum.get(g)).doubleValue();
System.out.println("kom hit %" + klasses.size() + " " + rowsum.size());
// test if rowsum = 0
for(int a = 0; a < rowsum.size(); a++)
    if(((Double) rowsum.get(a)).doubleValue() == 0.0)
        System.out.println("\ø\ strelsen er " + rowsum.size());
System.out.println("rowsum er null");
System.exit(-1);
}
for(int i = 0; i < size; i++)
    double c = 0.0;
    double d = 0.0;
    a1 = (HashMap) hasArray.get(i);
    a2 = (HashMap) hasArray.get(hasArray.size()-2);
    Set setting = at1.keySet();
    Iterator iter = setting.iterator();
while (iter.hasNext()) {
    Object o = iter.next();
    String s2 = o.toString();

    String value = (String) a1.get(o);
    double cc = Double.parseDouble(value);

    String a = (String) a2.get(s2);
    double dd = Double.parseDouble(a);

    double f = cc + dd;
    String ii = Double.toString(f);

    a2.put(s2, ii);
    hasjArray.set((hasjArray.size() - 2), a2);
}
System.out.println("kom hxt");
/*
String aaa = "";
a1 = (HashMap) hasjArray.get(hasjArray.size() - 2);
Collection collas = a1.values();
Iterator it = collas.iterator();
while (it.hasNext()) {
    Object o = it.next();
    String ss = o.toString();
    System.out.println(ss);
}
*/

for (int i = 0; i < size; i++) {
    //14
    a1 = (HashMap) hasjArray.get(i);
    a2 = (HashMap) hasjArray.get(hasjArray.size() - 1);
    a3 = (HashMap) hasjArray.get(hasjArray.size() - 2);
    Set setting = a1.keySet();
    Iterator iter = setting.iterator();

    while (iter.hasNext()) {
        Object o = iter.next();
        String s2 = o.toString();

        String value3 = (String) a1.get(o);
        double oo = Double.parseDouble(value3);

        //System.out.println("er her1 " + oo);
    }
String columnsumed = (String) a3.get(s2);

double columnsumed = Double.parseDouble(columnsumed);

//System.out.println("columnsumed " +columnsumed);
double e = ((columnsumed) * (Double) rowsum.get(i)).doubleValue()
/ total;
//System.out.println("e "+e);
double ny = (((oo-e) * (oo-e)))/e);

/*
if (Double.isNaN(ny))
System.out.println(s2 + " stoppa her");
*/

String valuem = (String) a2.get(o);
double yes = Double.parseDouble(valuem);
double uu = yes + ny;
a2.put(s2, Double.toString(uu));
hashArray.set((hashArray.size()-1),a2);
//has[j].put(s2, Integer.toString(ny));
}
}

a1 = (HashMap) hasjArray.get(hasjArray.size()-1);
Set setting = a1.keySet();
Iterator iter = setting.iterator();
while(iter.hasNext()){
    Object o = iter.next();
    String s2 = o.tostring();

    String value3 = (String) a1.get(o);
    //System.out.println(s2 + " trala");
    //double oo = Double.parseDouble(value3);
    //int aa = (int) oo;
    //a1.put(s2, Integer.toString(aa));
}

System.out.println("er der "+i);
int b = 0;
System.out.println("kalla");

List entrylist = new ArrayList(a1.entrySet());
/*
for(int a = 0; a < entrylist.size(); a++){
    Map.Entry e1 = (Map.Entry) entrylist.get(a);
    System.out.println(e1.getKey().toString() + " "+ e1.getValue());
*/
```java
public class Example {

    public static void main(String[] args) {
        try {
            File file = new File("example.txt");
            Scanner scanner = new Scanner(file);
            List<String> words = new ArrayList<>();
            while (scanner.hasNext()) {
                String line = scanner.nextLine();
                words.add(line);
            }
            scanner.close();

            Collections.sort(entryList, new ValueComparator());

            try {
                FileOut putStream out = new FileOutputStream("medstem.txt", true);
                PrintStream p = new PrintStream(out);
                List<String> sortedList = entryList.subList(entryList.size() - chi, entryList.size());
                chiwords = new String[chisize];
                //p.println("NY LIsT SIZE") + " JADA LISTA ER");

                for (int t = 0; t < antallet.length; t++) {
                    p.println(antallet[t] + " ");
                }
                p.println();
                /*
                 for (int i = nyList.size() - 1; i > -1; i--)
                 
                 Map.Entry e1 = (Map.Entry) nyList.get(i);
                 chiwords[i] = e1.getKey().toString();
                 System.out.println(e1.getKey().toString() + " " + e1.getValue());
                 */
                p.println(e1.getKey().toString() + " ");
                for (int o = 0; o < size; o++) {
                    Object aff = hasj[o].get(chiwords[i]);
                    String j = aff.toString();
                    p.println(j + " ");
                }
                p.println();
                /*
                 System.out.println("closing");
                 
                 //out.close();
                 System.out.println("NY LIsT SIZE") + " JADA LISTA ER");

                for (int t = 0; t < antallet.length; t++) {
                    totalen += antallet[t];
                }

                System.out.println("LENGTH er àm " + chiwords.length);
                // catch(Exceptio a) {System.out.println("noe gikk feil");}
                // System.out.println("bbaa + size");
            }
        } catch (Exception e) {
            System.out.println("Error: "+ e.getMessage());
        }
    }
}
```
```java
    hasj2 = new HashMap[size];
    for (int a = 0; a < size; a++)
        hasj2[a] = new HashMap(chiwords.length);
    
    /*
    Set settingq = hasj[1].keySet();
    Iterator iterq = settingq.iterator();
    while(iterq.hasNext()){
        Object o = iterq.next();
        String s2 = o.toString();
        Object s3 = hasj[2].get(s2);
        System.out.println(s2 + "&&&&&&&&& & & & & & & & & & & & & + s3);
    }
    */

    public void classify(){
        System.out.println("\østrresem er da " + size);
        int qq = 0;
        for(int a = 0; a < size; a++)
            double v = 1.0;
            System.out.println("HHHHHHHHHHHHHHHHH");
        for(int r = 0; r < chiwords.length; r++)
            if(test2.containsKey(chiwords[r])){
                System.out.println(chiwords[r]);
                QQ++;
                Object aff = hasj[a].get(chiwords[r]);
                String j = aff.toString();
                System.out.println(j);
                double g = Integer.parseInt(j);
                System.out.println(((String)klasses.get(a) + "\vikig\vikig\vikig\vikig\vikig" + g + "\vikig\vikig\vikig\vikig\vikig" + antallet[a] +"\vikig % % %" + v);
            }
            String s4 = (String) a1.get(chiwords[r]);
            double oo = Double.parseDouble(s4);
            System.out.println(oo);
            double k = v * oo;
            System.out.println(g + "YYYYY" + v + chiwords[r] + "a" + antallet[a]);
        hasj2[a].put(chiwords[r], Double.toString(v));
    }
    System.out.println("q = " + qq);
    for(int g = 0; g < size; g++)
        double y=1.0;
    if(g==0){
        System.out.println("Using chiwords:");
    }
    for(int h = 0; h < chiwords.length; h++)
```
Appendix A.1 Java Code

```java
if (test2.containsKey(chiwords[h])) {
    if (g >= 0) {
        System.out.println(chiwords[h] + " ");
    }
}

if (g == 0) {
    y = Double.parseDouble(hasj2[g].get(chiwords[h]).toString());
}

double k = y * (double) antallet[g];
System.out.println((String) klassen.get(g) + " " + k / totalen + " " + antallet[g]);

System.out.println("done");
```

```java
class ValueComparator implements Comparator {
    public int compare(Object o1, Object o2) {
        Map.Entry e1 = (Map.Entry) o1;
        Map.Entry e2 = (Map.Entry) o2;
        Object c1 = e1.getValue();
        Object c2 = e2.getValue();
        String c11 = c1.toString();
        String c22 = c2.toString();
        double a1 = Double.parseDouble(c11);
        double a2 = Double.parseDouble(c22);

        return Double.compare(a1, a2);
    }
}
```

Annotate.java

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.util.*;
```
import java.util.ArrayList;
import java.io.*;

public class Annotate{
    public Stem stem;
    public String ss2;
    public Pattern p;
    public Pattern splitter;
    public Matcher m;
    public int ss = 0;
    public ArrayList array.ba.goer;
    public FileInputStream fstream1;
    public ArrayList klasser.ja;
    public String[] oo;
    public int hh = 0;
    public String[] words;
    public ArrayList a;
    public int antall.chi_nr;
    public Jdbc jdb;
    public int till = 0;
    public String filename.go;
    public boolean stemn;
    public String[] hei.stopwords;
    public HashMap hashMap;
    public int[] antallet;
    public ArrayList test;
    public HashMap mappen;
    public Annotate(String s, int antall, String go, boolean stemer, int chi_nr){
        stemn = stemer;
        this.antall = antall;
        this.chi_nr = chi_nr;
        this.go = go;
        array = new ArrayList();
        a = new ArrayList();
        System.out.println("hoved");
        try {
            stem = new Stem();
            antallet = new int[14];
            hei = tokensem(s);
            stopwords = getStopWords();
            hashMap = new HashMap();
        }
        for(int i = 0; i < hei.length; i++){
            hashMap.put(hei[i],"0");
        }
        for(int i = 0; i < hei.length; i++){
            Object ai = hashMap.get(hei[i]);
            String b = ai.toString();
            int c = Integer.parseInt(b);
            c++;
            String h = Integer.toString(c);
            hashMap.put(hei[i].h);
        }
        System.out.println("kom høl øsj");
        if(go.equals("")){
```java
class J a v a Code
```
```java
private class Cache {
    void classify_cache() {
        }
        else {
            classify_with_ch() {
        }
        }
    }
    catch (Exception exception) {
        exception.printStackTrace();
    }
}

public String[] getStopWords() {
    try {
        FileInputStream in_stream = new FileInputStream("stopwords.txt");
        DataInputStream in = new DataInputStream(in_stream);
        String oo = new String[190];
        while (in.available() != 0) {
            oo[hh] = in.readLine();
        // System.out.println(oo[hh]);
            hh++;
        }
        in.close();
    }
    catch (Exception h) {
        return oo;
    }
}

public void classify_with_ch() {
    try {
        Jdbc jdbc = new Jdbc();
        Object a = jdbc.get_child(go);
        Object klasses = jdbc.get_chil(go);
        System.out.println(a.size());
    }
    Set settingr = hashMap.keySet();
    Iterator iterator = settingr.iterator();
    while (iterator.hasNext()) {
        Object o = iterator.next();
        String s2 = o.toString();
        String value3 = (String) hashMap.get(o);
        System.out.println(s2 + " + " + value3);
        System.out.println("starting parsing");
        Parse sosss = new Parse(a, jdbc, klasses);
        System.out.println("done parsing");
        ba = sosss.get_text();
        array = sosss.get_pubmed_id();
    }
```
goer = soss.get_go();

test = new ArrayList();
mappen = new HashMap();
ja = new ArrayList();

for (int i = 0; i < ba.size(); i++)
    String[] halo = tokenstem((String)ba.get(i));
    ja.add(halo);
    for (int ii = 0; ii < halo.length; ii++)
        mappen.put(halo[ii]."0";)
word_counter();
remove_stopwords();

Chicha chi = new Chicha(klasser, stem, mappen, hashMap, array, goer, jdbcb, stemn, chi_n);
Object o = iter.next();
s2 = o.toString();
String value3 = (String) mappen.get(o);
int hg = Integer.parseInt(value3);
// if (hg == 1){
// test.add(s2);
// }

ss++;
}

for (int i = 0; i < stopwords.length; i++){
    // System.out.println(stopwords[i]);
    mappen.remove(stopwords[i]);
}

for (int z = 0; z < test.size(); z++){
    String gg = (String) test.get(z);
    mappen.remove(gg);
}

public void classify_cache(){
    try{
        System.out.println("inner i cache");
        Jdbc jdbc = new Jdbc();
        // a = jdbc.get_rchild("GO:0003674");
        klasser = jdbc.get_child("GO:0003674");
        if (stemn){
            filename = "system.txt";
        }
        else{
            filename = "utenstem.txt";
        }
        FileInputStream fstream = new FileInputStream(filename);
        DataInputStream in = new DataInputStream(fstream);
        String[] chiwords = new String[chi_nr];
        StringTokenizer st = new StringTokenizer("this is a test");
        HashMap[] hasj = new HashMap[14];
        if (tell==0){
            int b = 0;
            String tu = in.readLine();
            StringTokenizer str = new StringTokenizer(tu);
            while (str.hasMoreTokens()) {
                int d = Integer.parseInt(str.nextToken());
                System.out.println(d);
                antallet[b] = d;
                b++;
            }
        }
    }
}
for(int a = 0; a < 14; a++){
    hasj[a] = new HashMap(chi_nr);
}

while (in.available() != 0 && tell < chi_nr)
{
    String tt = in.readLine();
    StringTokenizer sts = new StringTokenizer(tt);
    while (sts.hasMoreTokens()) {
        chiwords[tell] = sts.nextToken();
        for(int a = 0; a < 14; a++){
            hasj[a].put(chiwords[tell], sts.nextToken());
        }
        tell++;
    }
    in.close();
    int size = 14;
    HashMap[] hasj2 = new HashMap[size];
    for(int a = 0; a < size; a++){
        hasj2[a] = new HashMap(chiwords.length);
    }
    System.out.println("\nstr" + size);
}

int qq = 0;
double y = 1.0;
for(int a = 0; a < size; a++){
    double v = 1.0;
    y = 1.0;
    for(int r = 0; r < chiwords.length; r++){
        if(hasMap.containsKey(chiwords[r])){
            //System.out.println(chiwords[r]);
            qq++;
            Object aff = hasj[a].get(chiwords[r]);
            String j = aff.toString();
            //System.out.println(j);
            double g = Integer.parseInt(j);
            v = ((1.0 + g) * 100 / 1000 + antallet[a]);
            y *= v;
            //System.out.println(g + "YYYYY" + v + chiwords[r] + "" + antallet[a]);
            //hasj2[a].put(chiwords[r], Double.toString(v));
        }
        double k = y * (double) antallet[a];
        System.out.println("klasser.get(a) + " + k/26000.0 + " + antallet[a]);
/ *
 * System.out.println(“q = “ + qq);
 * for(int g = 0; g < size; g++){
 *   double y=1.0;
 *   if(g==0){
 *     System.out.println(“antall documents from medline: “ + antall);
 *     System.out.println(“Using chiwords:”);
 *   }
 *   for(int h = 0; h < chiwords.length; h++){
 *     if(hashMap.containsKey(chiwords[h])){
 *       if(g==0){
 *         System.out.print(chiwords[h] + “ “);
 *       }
 *     }
 *   }
 *   y = Double.parseDouble(hasj2[g].get(chiwords[h]).toString());
 *   double k = y * (double) antall[g];
 *   System.out.println((String) klasses.get(g) + “ " + k/26000.0 + “## " +antall[g]);
 *   //System.out.println((String) klasses.get(g) + “ “ + y);
 * }
 * }
 * catch (Exception exception) {
 *   exception.printStackTrace();
 * }
 * }
 *}

public String[] tokenstem(String streng){
  p = Pattern.compile(“\P(\p{C})” );
  m = p.matcher(streng);
  String juha = m.replaceAll(“”);
  //Pattern splitter = Pattern.compile ("[.:: ]++ “);
  Pattern splitter = Pattern.compile (”[ ]++ “);
  String[] words = splitter.split(juha);
  if(stemn){
    for(int i = 0; i < words.length; i++){
      words[i] = stem hei(words[i]);
    }
  }else{
    for(int i = 0; i < words.length; i++){
      words[i] = words[i].toLowerCase();
    }
  }
  return words;
}
```java
public static void main(String args[]) {
    boolean stem = false;
    if (args.length==4) {
        int tall = Integer.parseInt(args[0]);
        if (tall==1) {
            stem = true;
        }
    } String option = args[1];
    if (option.equals("-g")) {
        String searchfor = args[2];
        int chinumber = Integer.parseInt(args[3]);
        // String searchfor = "peg3 AND Parternally expressed gene";
        System.out.println("error");
        // Search a = new Search (searchfor,"",stem,chinumber);
        Search a = new Search(searchfor);
        String tekst = a.get_text();
        int antall = a.get_count();
        Annotate anno = new Annotate(tekst, antall, "", stem, chinumber);
    } if (option.equals("-t")) {
        String pubid = args[2];
        int chinumber = Integer.parseInt(args[3]);
        ArticleFetch fetch = new ArticleFetch(pubid);
        String article = fetch.get_article();
        // ArticleFetch test = new ArticleFetch(article);
        Annotate anno = new Annotate(article, 1,"",stem,chinumber);
    } else {
        System.out.println("wrong options, try -t or -g");
        System.exit(-1);
    }
}
```
A APPENDIX

A.1 Java Code

```
if (option.equals("-t")) {
    String pubid = args[2];
    int chinumber = Integer.parseInt(args[3]);
    String go = args[4];
    ArticleFetch fetch = new ArticleFetch(pubid);
    String article = fetch.get_article();
    Annotate anno = new Annotate(article, go, stem, chinumber);
}
else {
    System.out.println("Wrong options, try -t og -g");
    System.exit(-1);
}
else {
    System.out.println("Wrong number of arguments, must be at least 4");
    System.exit(-1);
}
```

```
public class Download extends DefaultHandler{
    public String aen, ben;
    FileOutputStream out; // declare a file output object
    PrintStream p; // declare a print stream object
    String pubmedid;
    String a, b;
    int tall;
    int restart = 0;
    int retmax = 500;
    int jada;
    String aleta;
    FileInputStream tstream;
    DataInputStream in;
}
```
public Download (){ try{
    fstream = new FileInputStream("aviss3.txt");
in = new DataInputStream(fstream);
    URL url;
    URLConnection urlConn;
    DataOutputStream printout;
    DataInputStream input;
    alfa = pubmedid;

    while(in.available() !=0){
        b = " ";
        for(int i = 0; i < 500; i++){
            a = in.readLine() + " . " ;
            b = b + a;
        }

        pubmedid = b.substring(0, b.length() -1 );
        if(pubmedid.equals("" )){
            System.out.println("er her");
            break;
        }

    }

    //System.out.println(pubmedid);
    System.out.println("restart er" + retstart);

db=pubmed&id="+ pubmedid +"&retmax=" + retmax + "&retmode=xm l";
    url = new URL (urlen);
    urlConn = url.openConnection();
    urlConn.setDoInput (true);
    urlConn.setDoOutput (true);
    urlConn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
    input = new DataInputStream (urlConn.getInputStream () );
    out = new FileOutputStream("myfile.xml",true);
    // Connect print stream to the output stream
    p = new PrintStream( out );

    System.out.println("ket");

    String str;
    while (null != (str = input.readLine())){
        p.println(str);
    }

    p.close();
    input.close();

    DateFormat df = DateFormat.getInstance();
    int seconds = df.getTimeInstance().get(Calendar.SECOND);
```java
import java.awt.*;
import java.io.*;
import java.util.*;
import java.util.Calendar;
import java.text.DateFormat;
import java.net.*;
import org.xml.sax.*;
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import java.util.ArrayList;

public class Download extends DefaultHandler{
    public String aen, ben;
    FileOutputStream out; // declare a file output object
    PrintStream p; // declare a print stream object
    String pubmedid;
    String a, b;
    int tall;
    int setstart = 0;
    int retmax = 500;
    int jada;
    String alfa;
    FileInputStream fstream;
    DataInputStream in;

    public Download (){ try {
        fstream = new FileInputStream("avis3.txt");
        in = new DataInputStream(fstream);
```

```java
System.out.println(seconds);
tall = df.getCalendar().get(Calendar.SECOND);
while (tall < (seconds + 2)) {
    DateFormat d = DateFormat.getTimeInstance();
tall = d.getCalendar().get(Calendar.SECOND);
}
System.out.println(a);
```
```java
URL url;
URLConnection urlConn;
DataOutputStream printout;
DataInputStream input;
alfa = pubmedid;

while (in.available() != 0)
    b = "";
    for (int i = 0; i < 500; i++)
        if (in.available() != 0)
            a = in.readLine() + ",";
        b = b + a;

pubmedid = b.substring(0, b.length() - 1);
if (pubmedid.equals(""))
    System.out.println("er her");
    break;

// System.out.println(pubmedid);
System.out.println("restart er" + retstart);

  db=pubmed&d" + pubmedid + "bremax=" + retmax + "bremode=xml"
;
url = new URL (urlen);
urlConn = url.openConnection();
urlConn.setDoInput (true);
urlConn.setDoOutput (true);
urlConn.setRequestProperty
("Content-Type", "application/x-www-form-urlencoded");
input = new DataInputStream (urlConn.getInputStream ());

out = new FileOutputStream ("myfile.sm1", true);
// Connect print stream to the output stream
p = new PrintStream (out);

System.out.println("her");

String str;
while (null != (str = input.readLine()))
    { p.println (str); }

p.close();
    input.close();

DateFormat df = DateFormat.getTimeInstance();
int seconds = df.getCalendar().get(Calendar.SECOND);
System.out.println(seconds);
tail = df.getCalendar().get(Calendar.SECOND);
while (tail < (seconds + 2))

DateFormat d = DateFormat.getTimeInstance();
```

A.1 Java Code
A. APPENDIX

A.1 Java Code

```java
94    tall = d.getCalendar().get(Calendar.SECOND);
95          System.out.println(a);
96          } //for
97
98          } catch (Exception e)
99          {
100             System.err.println("Exception: " + e);
101          }
102
103          } //end of constructor
104
105          public static void main(String args[]){
106          Test test = new Test();
107          }
108
109          } // End of class Happy.
```

Stem.java

```java
/*
Porter stemmer in Java. The original paper is in
no. 3, pp 130–137.
See also http://www.tartarus.org/~martin/PorterStemmer

History:

Release 1

Bug 1 (reported by Gonzalo Parra 16/10/99) fixed as marked below.
The words 'aerd', 'eed', 'oed' leave k at 'a' for step 3, and b[k–1] is then
outside the bounds of b.

Release 2

Similarly.

Bug 2 (reported by Steve Dyrdahl 22/2/00) fixed as marked below.
'ion' by itself leaves j = -1 in the test for 'ion' in step 5, and
b[j] is then outside the bounds of b.

Release 3

Considerably revised 4/9/00 in the light of many helpful suggestions
from Brian Goetz of Quiotix Corporation (brian@quiotix.com).

Release 4

*/

import java.io.*;
```

71
A.1 Java Code

```java
// Stemmer, implementing the Porter Stemming Algorithm
// The Stemmer class transforms a word into its root form. The input
// word can be provided a character at time (by calling add()), or at once
// by calling one of the various stem(something) methods.

public class Stem {
    private char[] b; // offset into b
    private int i, i_end; // offset to end of stemmed word
    private static final int INC = 50; // unit of size whereby b is increased

    public Stem() {
        b = new char[INC];
        i = 0;
        i_end = 0;
    }

    // Add a character to the word being stemmed. When you are finished
    // adding characters, you can call stem(VOID) to stem the word.
    public void add(char ch) {
        if (i == b.length) {
            char[] new_b = new char[i+INC];
            for (int c = 0; c < i; c++) new_b[c] = b[c];
            b = new_b;
        }
        b[i++] = ch;
    }

    // Adds wLen characters to the word being stemmed contained in a portion
    // of a char[] array. This is like repeated calls of add(char ch), but
    // faster.
    public void add(char[] w, int wLen) {
        if ((i+wLen) >= b.length) {
            char[] new_b = new char[i+wLen+INC];
            for (int c = 0; c < i; c++) new_b[c] = b[c];
            b = new_b;
        }
        for (int c = 0; c < wLen; c++) b[i++] = w[c];
    }

    // After a word has been stemmed, it can be retrieved by toString().
    // or a reference to the internal buffer can be retrieved by getResultBuffer
    // and getResultLength (which is generally more efficient.)
    public String toString() { return new String(b, 0, i_end); }

    // Returns the length of the word resulting from the stemming process.
    public int getResultLength() { return i_end; }
}
```
A APPENDIX

A.1 Java Code

```java
// * Returns a reference to a character buffer containing the results of
// the stemming process. You also need to consult getRes ultLength()
// to determine the length of the result.
*/

public char[] getResultBuffer() { return b; }

// * cons(i) is true <=> b[i] is a consonant. */

private final boolean cons(int i)
{
    switch (b[i])
    {
        case 'a': case 'e': case 'i': case 'o': case 'u': return false;
        case 'y': return (i==0) ? true : !cons(i-1);
        default: return true;
    }
}

// * m() measures the number of consonant sequences between 0 and j. If c is
// a consonant sequence and v a vowel sequence, and <..> indicates arbitrary
// presence.

private final int m()
{
    int n = 0;
    int i = 0;
    while(true)
    {
        if (i > j) return n;
        if (! cons(i)) break; i++;
    }
    i++;
    while(true)
    {
        if (i > j) return n;
            if (cons(i)) break;
        i++;
        n++;
        while(true)
        {
            if (i > j) return n;
                if (i cons(i)) break;
            i++;
        }
        i++;
    }
}

// * vowelin stem() is true <=> 0 ....j contains a vowel */

private final boolean vowelin stem()
{
    int i; for (i = 0; i <= j; i++) if (i cons(i)) return true;
    return false;
}

// * doublec(j) is true <=> j..(j-1) contain a double consonant. */
```
private final boolean doublec(int j) {
    if (j < 1) return false;
    if (b[j] != b[j-1]) return false;
    return cons(j);
}

/* cvc(i) is true <e> i-2,i-1,i has the form consonant – vowel – consonant
   and also if the second c is not w,x or y. this is used when trying to
   restore an e at the end of a short word. e.g.
   cav(e), luv(e), hop(e), crim(e), but
   snow, box, tray.

private final boolean cvc(int i) {
    if (i < 2 || cons(i) || cons(i-1) || !cons(i-2)) return false;
    if (ch == 'w' || ch == 'x' || ch == 'y') return false;
}

private final boolean ends(String s) {
    int l = s.length();
    int o = k-l+1;
    if (o < 0) return false;
    for (int i = 0; i < l; i++) if (b[o+i] != s.charAt(i)) return false;
    j = k-1;
    return true;
}

/* setto(s) sets (j+1) ....k to the characters in the string s, readjusting
   k. */

private final void setto(String s) {
    int l = s.length();
    int o = j+1;
    for (int i = 0; i < l; i++) b[o+i] = s.charAt(i);
    k = j+1;
}

/* r(s) is used further down. */

private final void r(String s) {
    if (m().length() > 0) setto(s);
}

/* step1() gets rid of plurals and –ed or –ing. e.g.
   caresses -> caress
   ponies -> poni
   ties -> ti
   caress -> caress
   cats -> cat
   feed -> feed
   agreed -> agree
   disabled -> disable
   matting -> mat
   mating -> mate
   meeting -> meet
   milling -> mill
private final void step1()
{
  if (b[k] == 's')
  { 
    if (ends("ssed")) k--; else
    if (ends("ses")) setto("s"); else
    if (b[k-1] == 's') k--; 
  }
  if (ends("sed")) { if (m() > 0) k--; else
  if (ends("ed") || ends("sing")) & vowelinstem())
  { k = j;
    if (ends("ate")) setto("ate"); else
    if (ends("bie")) setto("bie"); else
    if (ends("ise")) setto("ise"); else
    if (doublec(k))
    { k--; 
      if (ch == 'i' || ch == 's' || ch == 'z') k++;
    }
  } else if (m() == 1 & cvc(k)) setto("e");
}
}

/* step2() turns terminal y to i when there is another vowel in the stem. */
private final void step2()
{
  if (ends("y") & vowelinstem()) b[k] = 'i';
}

/* step3() maps double suffixes to single ones. so -ization (= -ize plus -ation) maps to -ize etc. note that the string before the suffix must give m() > 0. */
private final void step3()
{ if (k == 0) return; /* For Bug 1 */ switch (b[k - 1])
{
  case 'a': if (ends("ational")) { r("ate"); break; }
  if (ends("tional")) { r("tion"); break; }
  break;
  case 'c': if (ends("ence")) { r("ence"); break; }
  if (ends("enc")) { r("ence"); break; }
  break;
  case 'e': if (ends("ise")) { r("ise"); break; }
  break;
  case 'i': if (ends("bie")) { r("bie"); break; }
  if (ends("ai")) { r("ai"); break; }
  if (ends("enti") | r("ent"); break; }
  if (ends("el") | r("el"); break; }
  if (ends("ousi") | r("ous"); break; }
  break;
  case 'o': if (ends("ousion") | r("ous"); break; }
  if (ends("ation") | r("ate"); break; }
  if (ends("ator") | r("ate"); break; }
  break;
  case 's': if (ends("aitism") | r("ai"); break; }
  if (ends("suesse") | r("sue"); break; }
  if (ends("finesse") | r("fine"); break; }
  if (ends("ousness") | r("ous"); break; }
  break;
}
case 't': if (ends("alite")) { r("al"); break; } 
    if (ends("uwut")) { r("two"); break; } 
    if (ends("blite")) { r("bie"); break; } 
    break;
  case 'g': if (ends("logd")) { r("log"); break; } 
} 
/ * step4() deals with -ic, -full, -ness etc. similar strategy to step3. */

private final void step4() { switch (b[k]) 
  { 
    case 'e': if (ends("case")) { r("sc"); break; } 
      if (ends("active")) { r("ac"); break; } 
      if (ends("alize")) { r("al"); break; } 
      break;
    case 's': if (ends("scsit")) { r("sc"); break; } 
      break;
    case 'l': if (ends("cal") ) { r("sc") ; break; } 
      if (ends("ful")) { r("sn") ; break; } 
      break;
    case 's': if (ends("mess")) { r("mn") ; break; } 
      break;
  } 
/ * step5() takes off -ant, -ence etc., in context <C>vc<Y>. */

private final void step5() 
  { if (k == 0) return; / * for Bug 1 */ switch (b[k-1]) 
    { 
      case 'a': if (ends("ai")) break; return;
      case 'c': if (ends("ance")) break;
      if (ends("emce")) break; return;
      case 'e': if (ends("ene") ) break; return;
      case 'i': if (ends("ic") ) break; return;
      case 'l': if (ends("abile")) break;
      if (ends("stble")) break; return;
      case 'n': if (ends("ant") ) break;
      if (ends("emnt") ) break;
      if (ends("ment") ) break;
      / * element etc. not stripped before the m */
      if (ends("emnt") ) break; return;
    }
    case 'o': if (ends("om") ) break; return;
      if (ends("oam") ) break;
      / * takes care of -ous */
      case 's': if (ends("sam") ) break; return;
    case 't': if (ends("aste") ) break;
    if (ends("stse") ) break; return;
    case 'u': if (ends("uos") ) break; return;
    case 'v': if (ends("svc") ) break; return;
    case 'z': if (ends("zce") ) break; return;
    default: return;
  } 
  if (m() > 1) k = j;
}

/ * step6() removes a final -e if m() > 1. */

private final void step6() 
  { j = k;
    if (b[k] == 'e')
      { int a = m();
        / * j = k, */
      ....
if (a > 1) || a == 1 && !cvc(k-1)) k--; 
}
if (b[k] == 'l' && doublec(k) && m() > 1) k--; 
}

/** Stem the word placed into the Stemmer buffer through calls to add().
 * Returns true if the stemming process resulted in a word different
 * from the input. You can retrieve the result with
 * getResultSet()/getResultBuffer() or toString().
 */
public void stem()
{
    k = i - 1;
    if (k > 1) { step1(); step2(); step3(); step4(); step5(); step6(); }
    i = k+1; i = 0;
}

/** Test program for demonstrating the Stemmer. It reads text from a
 * a list of files, stems each word, and writes the result to standard
 * output. Note that the word stemmed is expected to be in lower case:
 * forcing lower case must be done outside the Stemmer class.
 * Usage: Stemmer file-name file-name ...
 */
public String hei(String hallo){
    String a = hallo.toLowerCase();
    char[] b = a.toCharArray();
    for(int i = 0; i < b.length; i++){
        if (!Character.isLetter(b[i])){
            return a;
        }
    }
    for(int i = 0; i < b.length; i++){
        if (Character.isLetter(b[i])){
            add(b[i]);
        }
    }
    stem();
    String u;
    u = toString();
    return u;
}
## B Manual annotations for the test genes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>GO id</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADH1</td>
<td>GO:0004022</td>
<td>alcohol dehydrogenase activity</td>
</tr>
<tr>
<td>ADH1</td>
<td>GO:0016491</td>
<td>oxidoreductase activity</td>
</tr>
<tr>
<td>ADH1</td>
<td>GO:0004024</td>
<td>alcohol dehydrogenase activity, zinc-dependent</td>
</tr>
<tr>
<td>ARSB</td>
<td>GO:0003943</td>
<td>N-acetylgalactosamine-4-sulfatase activity</td>
</tr>
<tr>
<td>ARSB</td>
<td>GO:0016787</td>
<td>hydrolase activity</td>
</tr>
<tr>
<td>ARSB</td>
<td>GO:0008484</td>
<td>sulfuric ester hydrolase activity</td>
</tr>
<tr>
<td>CACNA1A</td>
<td>GO:0004245</td>
<td>voltage-gated calcium channel activity</td>
</tr>
<tr>
<td>CD58</td>
<td>GO:0005194</td>
<td>cell adhesion molecule activity</td>
</tr>
<tr>
<td>CD58</td>
<td>GO:0005102</td>
<td>receptor binding activity</td>
</tr>
<tr>
<td>CDS1</td>
<td>GO:0004605</td>
<td>phosphatidate cytidylyltransferase activity</td>
</tr>
<tr>
<td>CDS1</td>
<td>GO:0004142</td>
<td>diacylglycerol cholinephosphotransferase activity</td>
</tr>
<tr>
<td>CDS1</td>
<td>GO:0000287</td>
<td>magnesium ion binding activity</td>
</tr>
<tr>
<td>CDS1</td>
<td>GO:0007165</td>
<td>signal transduction</td>
</tr>
<tr>
<td>CR16</td>
<td>GO:0017124</td>
<td>SH3-domain binding activity</td>
</tr>
<tr>
<td>CR16</td>
<td>GO:0008651</td>
<td>actin polymerizing activity</td>
</tr>
<tr>
<td>CR16</td>
<td>GO:0005522</td>
<td>profilin binding activity</td>
</tr>
<tr>
<td>JAG2</td>
<td>GO:0005112</td>
<td>Notch binding activity</td>
</tr>
<tr>
<td>JAG2</td>
<td>GO:0008083</td>
<td>growth factor activity</td>
</tr>
<tr>
<td>KCNK2</td>
<td>GO:0015271</td>
<td>outward rectifier potassium channel activity</td>
</tr>
<tr>
<td>KCNK2</td>
<td>GO:0005267</td>
<td>potassium channel activity</td>
</tr>
<tr>
<td>LGALS3</td>
<td>GO:0019863</td>
<td>IgE binding activity</td>
</tr>
<tr>
<td>LGALS3</td>
<td>GO:0016936</td>
<td>galactoside binding activity</td>
</tr>
<tr>
<td>LGALS3</td>
<td>GO:0008248</td>
<td>pre-mRNA splicing factor activity</td>
</tr>
<tr>
<td>LGALS3</td>
<td>GO:0008189</td>
<td>apoptosis inhibitor activity</td>
</tr>
<tr>
<td>LGALS3</td>
<td>GO:0005529</td>
<td>sugar binding activity</td>
</tr>
<tr>
<td>NKX2</td>
<td>GO:0015293</td>
<td>symporter activity</td>
</tr>
<tr>
<td>NKX2</td>
<td>GO:0015297</td>
<td>antiporter activity</td>
</tr>
<tr>
<td>PAM</td>
<td>GO:0005507</td>
<td>copper ion binding activity</td>
</tr>
<tr>
<td>PAM</td>
<td>GO:0005489</td>
<td>electron transporter activity</td>
</tr>
<tr>
<td>PAM</td>
<td>GO:0004504</td>
<td>peptidyl-fucine monoxygenase activity</td>
</tr>
<tr>
<td>PAM</td>
<td>GO:0004497</td>
<td>monoxygenase activity</td>
</tr>
<tr>
<td>PAM</td>
<td>GO:0016842</td>
<td>amidine-lyase activity</td>
</tr>
<tr>
<td>PAM</td>
<td>GO:0016715</td>
<td>oxidoreductase activity</td>
</tr>
<tr>
<td>PAM</td>
<td>GO:0016829</td>
<td>lyase activity</td>
</tr>
<tr>
<td>PEG3</td>
<td>GO:0003677</td>
<td>DNA binding activity</td>
</tr>
<tr>
<td>PEG3</td>
<td>GO:0003700</td>
<td>transcription factor activity</td>
</tr>
</tbody>
</table>
### C Result output

<table>
<thead>
<tr>
<th>GO:0003774</th>
<th>0.3693461538461538</th>
<th>97</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0003824</td>
<td>3894.768</td>
<td>10062</td>
</tr>
<tr>
<td>GO:0004871</td>
<td>283.19546153846153</td>
<td>2713</td>
</tr>
<tr>
<td>GO:0005198</td>
<td>7.702153846153847</td>
<td>447</td>
</tr>
<tr>
<td>GO:0005215</td>
<td>422.66215384615384</td>
<td>3312</td>
</tr>
<tr>
<td>GO:0005488</td>
<td>2159.7129230769233</td>
<td>7491</td>
</tr>
<tr>
<td>GO:0005554</td>
<td>0.6350769230769231</td>
<td>128</td>
</tr>
<tr>
<td>GO:0016209</td>
<td>0.21346153846153845</td>
<td>74</td>
</tr>
<tr>
<td>GO:0030188</td>
<td>0.002769230769230769</td>
<td>8</td>
</tr>
<tr>
<td>GO:0030234</td>
<td>27.625230769230768</td>
<td>847</td>
</tr>
<tr>
<td>GO:0030528</td>
<td>91.15615384615384</td>
<td>1539</td>
</tr>
<tr>
<td>GO:0030533</td>
<td>0.002769230769230769</td>
<td>8</td>
</tr>
<tr>
<td>GO:0045182</td>
<td>0.4696153846153846</td>
<td>110</td>
</tr>
<tr>
<td>GO:0045735</td>
<td>2.3076923076923076E-4</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table 10**: java -Xmx500m Annotate 1 -g LGALS3 100
<table>
<thead>
<tr>
<th>GO-id</th>
<th>Classification probability</th>
<th>Documents in subtree</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:000170</td>
<td>2.891302408644853E22</td>
<td>2</td>
</tr>
<tr>
<td>GO:0003826</td>
<td>1.1046997714076678E35</td>
<td>3</td>
</tr>
<tr>
<td>GO:0004033</td>
<td>2.339292758016471E93</td>
<td>15</td>
</tr>
<tr>
<td>GO:0004154</td>
<td>128.57399103139014</td>
<td>1</td>
</tr>
<tr>
<td>GO:0004497</td>
<td>7.267135278493212E180</td>
<td>191</td>
</tr>
<tr>
<td>GO:0004800</td>
<td>1.4001102045494227E92</td>
<td>9</td>
</tr>
<tr>
<td>GO:0009974</td>
<td>0.8609865470852018</td>
<td>1</td>
</tr>
<tr>
<td>GO:0010242</td>
<td>9.035320027015166E20</td>
<td>2</td>
</tr>
<tr>
<td>GO:0015002</td>
<td>1.092301361937848E125</td>
<td>40</td>
</tr>
<tr>
<td>GO:0015036</td>
<td>3.6625768060428334E88</td>
<td>14</td>
</tr>
<tr>
<td>GO:0016614</td>
<td>1.4163530530689861E182</td>
<td>207</td>
</tr>
<tr>
<td>GO:0016627</td>
<td>2.8849852764388115E145</td>
<td>72</td>
</tr>
<tr>
<td>GO:0016638</td>
<td>9.10458800684152E129</td>
<td>43</td>
</tr>
<tr>
<td>GO:0016645</td>
<td>9.6231811008365851E122</td>
<td>35</td>
</tr>
<tr>
<td>GO:0016651</td>
<td>5.479690462510348E167</td>
<td>109</td>
</tr>
<tr>
<td>GO:0016661</td>
<td>2.215706513744332E56</td>
<td>5</td>
</tr>
<tr>
<td>GO:0016667</td>
<td>3.0141022029931454E133</td>
<td>51</td>
</tr>
<tr>
<td>GO:0016675</td>
<td>1.092301361937848E125</td>
<td>40</td>
</tr>
<tr>
<td>GO:0016679</td>
<td>3.5212370504154435E134</td>
<td>30</td>
</tr>
<tr>
<td>GO:0016684</td>
<td>4.1556437986879551E140</td>
<td>45</td>
</tr>
<tr>
<td>GO:0016701</td>
<td>3.6273511127974754E137</td>
<td>46</td>
</tr>
<tr>
<td>GO:0016705</td>
<td>4.2847366404192151E184</td>
<td>181</td>
</tr>
<tr>
<td>GO:0016721</td>
<td>3.0786350065164925E110</td>
<td>19</td>
</tr>
<tr>
<td>GO:0016722</td>
<td>3.099879407369051E85</td>
<td>12</td>
</tr>
<tr>
<td>GO:0016725</td>
<td>2.2911885201793725E85</td>
<td>10</td>
</tr>
<tr>
<td>GO:0016730</td>
<td>8.2098883184315271E42</td>
<td>3</td>
</tr>
<tr>
<td>GO:0016903</td>
<td>1.41964172607434931E163</td>
<td>96</td>
</tr>
<tr>
<td>GO:0030611</td>
<td>1.610666557804675654E54</td>
<td>4</td>
</tr>
<tr>
<td>GO:0030613</td>
<td>1.9230316691397834E45</td>
<td>4</td>
</tr>
<tr>
<td>GO:0051213</td>
<td>6.523889514334321E139</td>
<td>48</td>
</tr>
</tbody>
</table>

*Table 11: java -Xmx500m Annotate 1 -g ADH1 400 GO:0016491*
### C RESULT OUTPUT

<table>
<thead>
<tr>
<th>GO-id</th>
<th>Classification probability</th>
<th>Documents in subtree</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0003774</td>
<td>37.29276923076923</td>
<td>97</td>
</tr>
<tr>
<td>GO:0003824</td>
<td>4.1225917266E7</td>
<td>10062</td>
</tr>
<tr>
<td>GO:0004871</td>
<td>798044.8106153846</td>
<td>2713</td>
</tr>
<tr>
<td>GO:0005198</td>
<td>3550.692923076923</td>
<td>447</td>
</tr>
<tr>
<td>GO:0005215</td>
<td>1462317.4246153845</td>
<td>3312</td>
</tr>
<tr>
<td>GO:0005488</td>
<td>1.6964336991461538E7</td>
<td>7491</td>
</tr>
<tr>
<td>GO:0005554</td>
<td>83.83015384615385</td>
<td>128</td>
</tr>
<tr>
<td>GO:0016209</td>
<td>16.65</td>
<td>74</td>
</tr>
<tr>
<td>GO:0030188</td>
<td>0.024923076923076923</td>
<td>8</td>
</tr>
<tr>
<td>GO:0030234</td>
<td>24144.45169230769</td>
<td>847</td>
</tr>
<tr>
<td>GO:0030528</td>
<td>147490.65692307692</td>
<td>1539</td>
</tr>
<tr>
<td>GO:0030533</td>
<td>0.024923076923076923</td>
<td>8</td>
</tr>
<tr>
<td>GO:0045182</td>
<td>54.47538461538461</td>
<td>110</td>
</tr>
<tr>
<td>GO:0045735</td>
<td>9.230769230769231E-4</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table 12:** java -Xmx500m Annotate 1 -t 9288754 4000