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 χ^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 χ^2 analysis (distribution). With very few experience available to build on, my task was rightfully expected to be a challange 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

List of Algorithms

List of Figures

List of Tables

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 hetrogenous 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, a 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 χ^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 necleus 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 structures 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 reffered as the DNA double helix structure. The two DNA chains run in opposite direction, one runs in what is called 5' $\rightarrow$ 3' direction while the other runs in 3' $\rightarrow$ 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 paring rules.

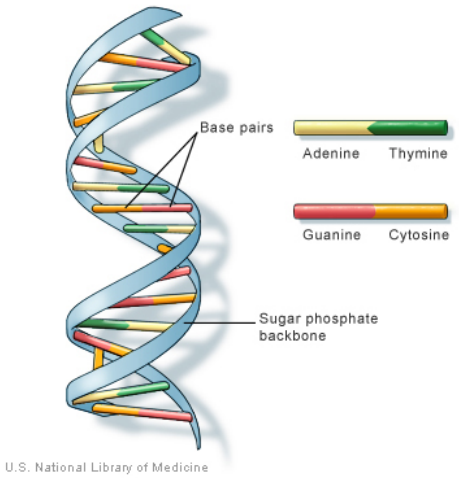

Figure 1: The figure shows the structure of the DNA double helix. The connection between A to T, and G to C can be seen. The figure is retrieved from the U.S National Library of Medicine's web pages.

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

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

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 acylic 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.

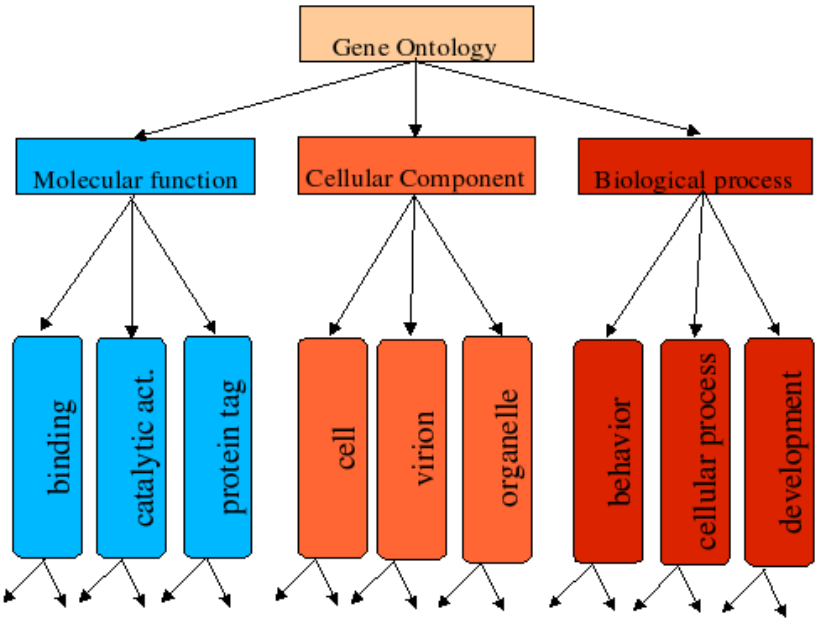

Figure 2: The Gene Ontology structure

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.

$$Precision = \frac{|\ Ra\ |}{|\ A\ |} \tag{1}$$

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

$$Recall = \frac{|\ Ra\ |}{|\ R\ |} \tag{2}$$

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 spesific 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.

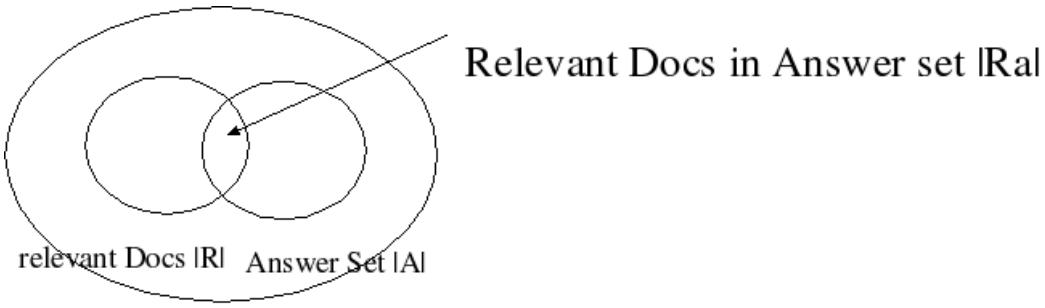

Figure 3: Precision and recall

2.2.2 Stemming

A word can have may variants e.g plurals, gerund forms and past tense suffixes. Many times it is desirable to remove such suffixes and treat every occurrences 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. Paralleling 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:

$$P[Y|X] = \frac{P[X|Y]P[Y]}{P[X]} \tag{3}$$

outlook	*temperature*	*humidity*	*windy*	*play*
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mil	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 2: Weather data

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 $C_1, C_2, C_3 ... C_m$, 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:

$$P(C_i|X) > P(C_j|X) \qquad \text{for all } j \text{ such that } \leq j \leq m, j \neq i. \tag{4}$$

2.3.2 Finding the class prior prababilities

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

$$P(yes) = P(C_1) = \frac{9}{14} \tag{5}$$

$$P(yes) = P(C_2) = \frac{5}{14} \tag{6}$$

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 claw conditional independence is made, i.e that for each class, the attributes are assumed to be independent. This assumption allows us to write

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

Let ut 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:

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

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 preposisitons 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 tockenization (i.e splitting the text into words) is difficult because of the frequently use of parentheses.

χ^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.

	RNA	*DNA*	*helicase*	*p68*	*Total*
GO:1	12	1	2	1	16
GO:2	2	12	2	1	13
GO:3	2	1	9	1	13
Total	16	14	13	3	48

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 table4 shows the calculated expectation values:

	RNA	*DNA*	*helicase*	*p68*
GO:1	5,6	4,9	4,5	1,0
GO:2	5,9	5,2	4,8	1,1
GO:3	4,5	3,9	3,7	0,8

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:

$$((O - E)2/E) \tag{9}$$

Where O is the observed values from table 5 and E is the expectation values from table4. 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:

	RNA	*DNA*	*helicase*	*p68*
GO:1	7,4	3,1	1,41	0,02
GO:2	2,6	9,0	1,6	0,01
GO:3	1,4	2,2	7,7	0,03
Total	11,4	14,3	10,7	0,06

Table 5: χ^2 values

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

2.3.5 Chi square used in genetics

The χ^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)

	AB	Ab	aB	ab
AB	AABb	AABb	AaBB	AaBb
Ab	AABb	AAbb	AaBb	Aabb
aB	AaBB	AaBb	aaBB	aaBb
ab	AaBb	Aabb	aaBb	aabb

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:

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

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:

$$P(GO:1|d) = 3/9 \times P(d|GO:1)$$
$$= P(d|GO:1) = P(RNA|GO:1) \times P(DNA|GO:1) \times P(helicase|GO:1)$$
$$= 3/9 \times 1 \times 0,41 \times 0,8 = 0,108$$

PMID	GO-node	RNA	P(RNA|GO)	DNA	P(DNA|GO)	helicase	P(helicase|GO)
111	GO:1	X	1,0	X	0,41	X	0,8
222	GO:1	X	-	0	-	0	-
333	GO:1	X	-	0	-	X	-
333	GO:2	X	0,41	X	0,41	X	0,8
444	GO:2	0	-	0	-	X	-
555	GO:2	0	-	X	-	0	-
666	GO:3	0	0,41	X	0,8	X	1,0
777	GO:3	0	-	X	-	X	-
888	GO:3	X	-	0	-	X	-

Table 7: In the table we see a training set of 8 articles, each belonging to one or more classes (here represented as GO-nodes). An *X* marks that the word is present in that certain document, whereas a *0* marks no occurrences. Using the equation 10 with s as 0,4 we get the probabilities of each word belonging each of the classes

$$P(GO:2|d) = 3/9 \times P(d|GO:2)$$
$$= P(d|GO:2) = P(RNA|GO:2) \times P(DNA|GO:2) \times P(helicase|GO:2)$$
$$= 3/9 \times 0,41 \times 0,41 \times 0,8 = 0,04$$

$$P(GO:3|d) = 3/9 \times P(d|GO:3)$$
$$= P(d|GO:3) = P(RNA|GO:3) \times P(DNA|GO:3) \times P(helicase|GO:3)$$
$$= 3/9 \times 0,41 \times 0,8 \times 1 = 0,108$$

Thus we can conclude that in this situation there is a larger probability that the document belongs to the class GO:2 than the others.

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 tranfering information. The advantage with XML is that it is easy to understand, process and generate. HTML (Hyper Text Markup Language) is also a instance of SGML, but can not be expanded with taggs like XML. First and foremost HTML is meant to show data using a web browser whereas XML is more about structuring the data.

```
<?xml version="1.0"?>

<doc>
<para>Hello, world!</para>
</doc>
```

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 unefficent 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:

start document
start element: doc
start element: para
characters: Hallo, verden!
end element: para
end element: doc
end document

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()

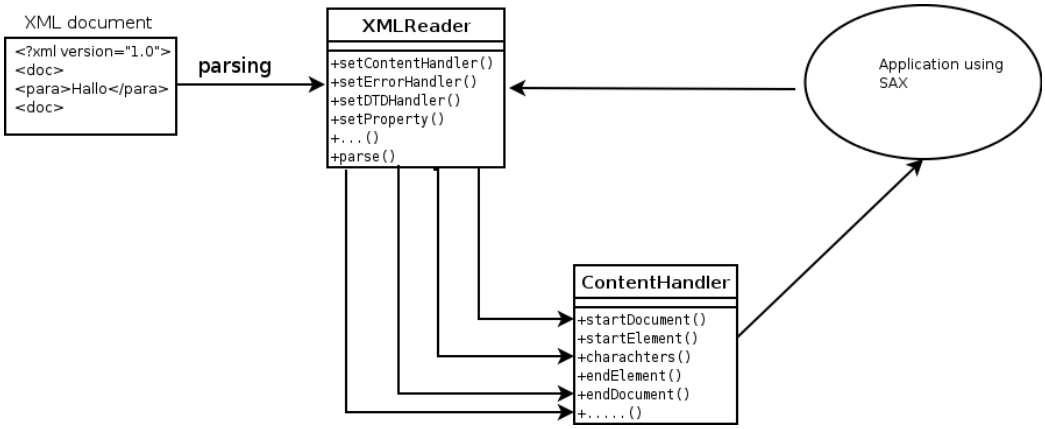

Figure 4: Parsing ved hjelp av SAX

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 ontologys. 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.

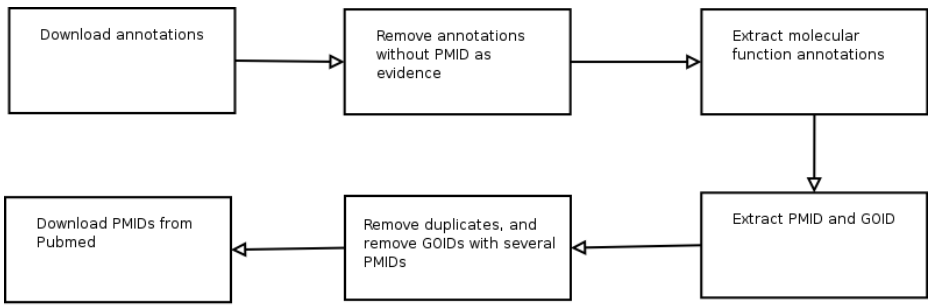

Figure 5: Overview of the steps involved in pre-processing

4.1 GO and mysql

The complete GO-hierarchy was downloaded from the website of Gene Ontology. At current time there are 17925 terms, 9395 biological process, 1550 cellular component and 6980 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 include 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 doucument classifier.

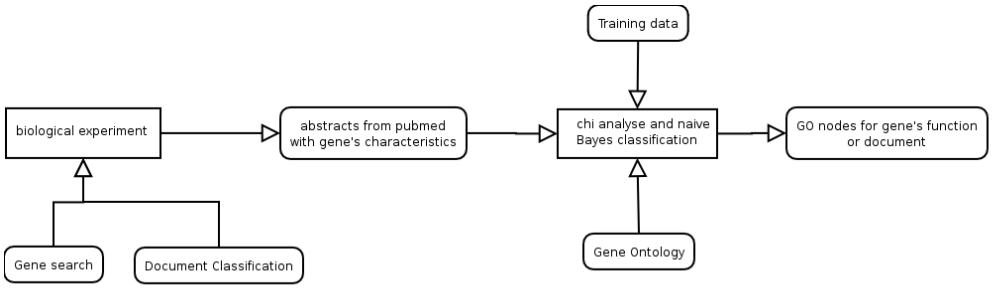

Figure 6: Overview of approach

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 χ^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 χ^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 a easy improvement, and we are more interesting in whether or not the methods we are introducing are working. An outline of our method is given in pseudo code in algorithm 1.

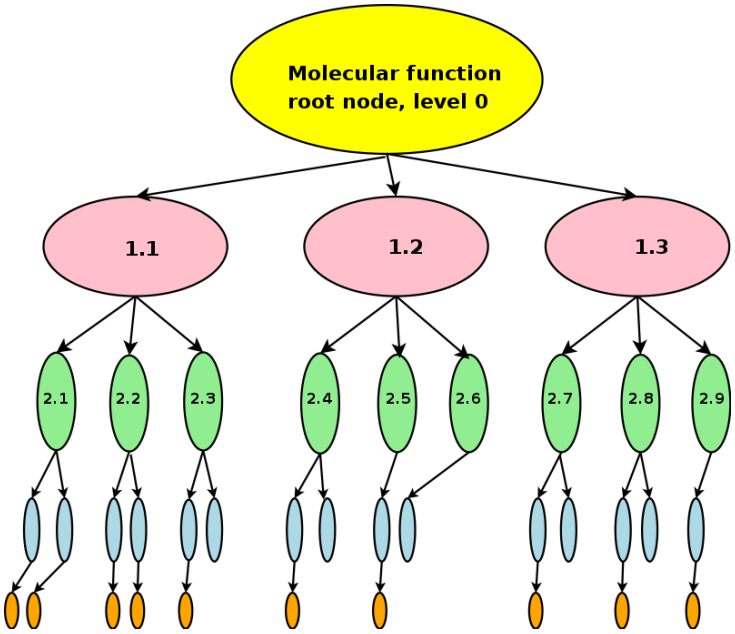

Figure 7: The subtree natur of the go-graph

5.2 System overview

We will give a short overview of the different processes involved in our system. In 8 the different processes are represented as boxes, and a short description of their methods and tasks will be given below.

- *Annotate* is the 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 algorihm shows the classification at just one level, the user must based on the result, start the classification on the next level

1: $d \Leftarrow$ words in document
2: $chi\_nr \Leftarrow 1000$
3: **for** $i = 0$ to number_of_subtrees **do**
4: $HashTable[i] \Leftarrow$ Words as keys, frequencies as values
5: Find χ-$scores$ for all the words
6: $P(w \mid i)$ {Using formula 10}
7: **end for**
8: $\chi\_array \Leftarrow$ Select chi_nr with highest χ-$scores$
9: **for** $i = 0$ to number_of_subtrees **do**
10: $chi\_hit \Leftarrow$ words that are both in document and in chi_array
11: **print** $P(c_i \mid d)$ {Using formula 8}
12: **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 getrchild returns the complete subtree of a node. And the method getancestor returns all

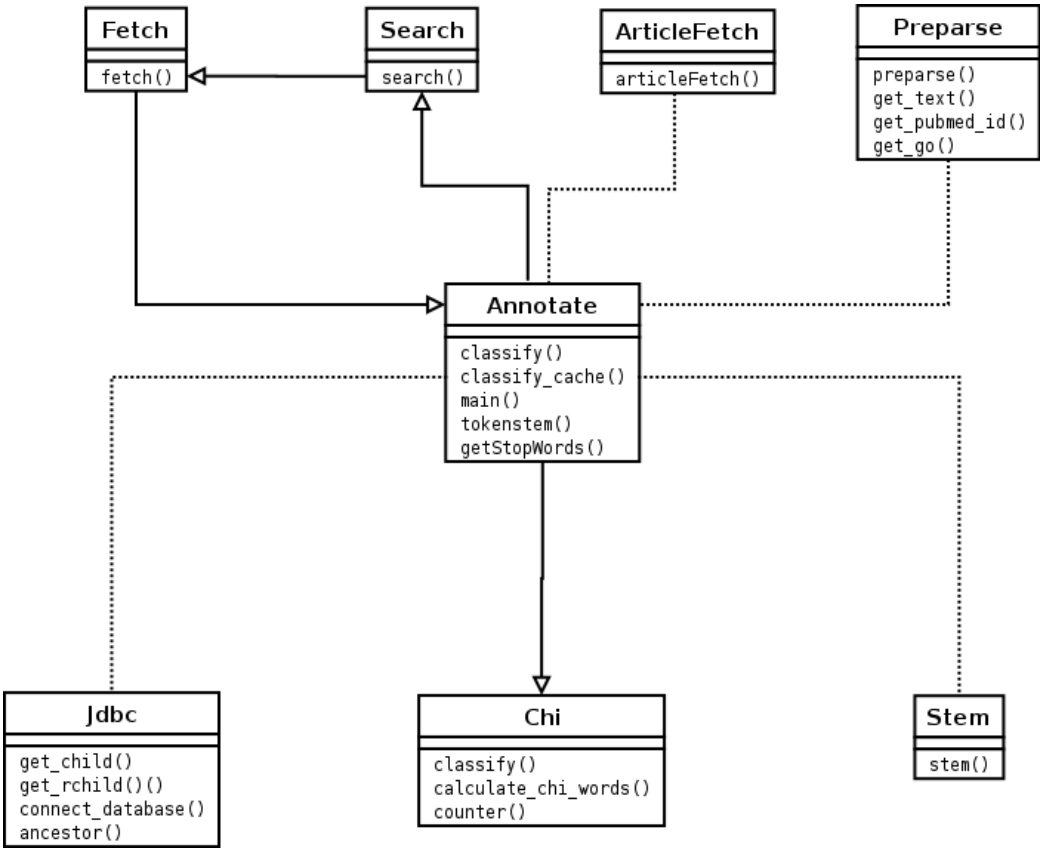

Figure 8: Class diagram

the ancestors to a node.

- *Chi* is the class that counts the occurrences of words in a subtree and performs the χ^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.

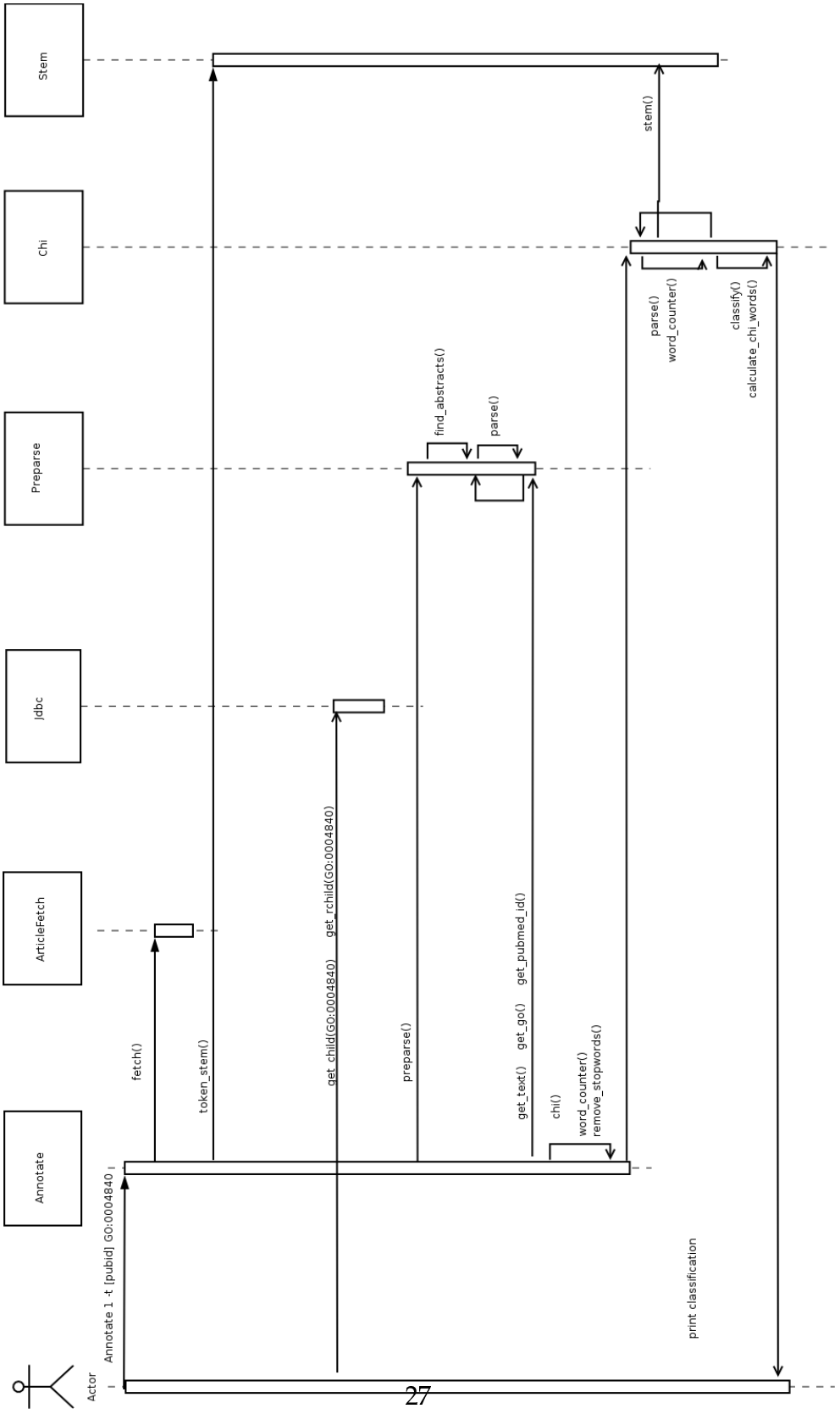

Figure 9: A typically usage of the system, showing the flow of the information from the user starts the program at the command line to the output of the classifications

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 Jana 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 algorihm 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 ⇐ 1
3: end if
4: if option nr 2 equals -g then
5:    gene_name ⇐ 2.option
6:    nr_of_chi ⇐ 3.option
7:    go ⇐ 4.option
8:    abstracts ⇐ Search.get_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 ⇐ 2.option
13:    nr_of_chi ⇐ 3.option
14:    go ⇐ 4.option
15:    abstract ⇐ ArticleFetch.get_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 ⇐ construct hash table of words in abstracts
4: for i = 0 to number_of_classes do
5:    prob_word_in_class ⇐ 1
6:    for j = 0 to χ-words.length do
7:       if words_abstracts contains χ-words j then
8:          number ⇐ class i, χ − word j
9:          number ⇐ ((1+number)×1000) / (1000 + number of documents in i)
10:         prob_word_in_class = prob_word_in_class × number
11:      end if
12:   end for
13:   classify ⇐ (classify × number of documents in i) / number of documents in total
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 χ^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.

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

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.

GO-id	Classification probability	Documents in subtree
GO:0004457	9.2739961651554O3E75	18
GO:0016615	1.41055495170629 77E72	24
GO:0016616	4.59026658710352 06E119	182
GO:0016898	2.425728858640503E46	8
GO:0016899	4.457844881615237E42	5

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 describtion of how we would have measured the precition 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:

$$HP_i = \frac{TP_i}{TP_i + FP_i} \qquad hR_i = \frac{TP_i}{TP_i + FN_i} \tag{11}$$

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*

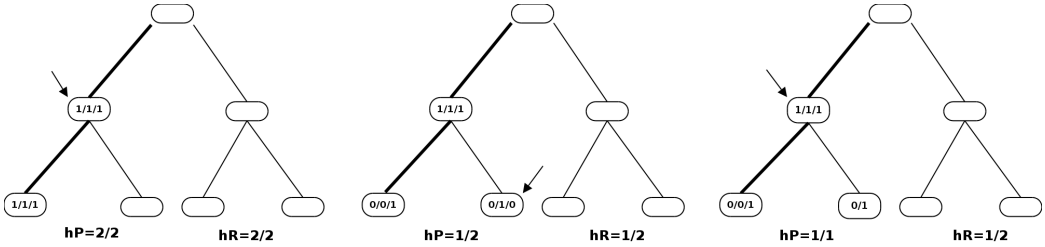

Figure 10: The ellipse at the bottom left represent the real category of a test instance, whereas the arrow is pointing to the category found by our classifier. The path from the root to the correct category is shown in bold. Numbers in the nodes represent the number of instances correctly assigned to 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 hown 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 treshold 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 χ^2 feature selection does not give good enough results. If a χ^2 word with high value is not biologically correct it effects the results drastically. We decided to keep all numbers as possible χ^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 out system will become better in the future.

8.5 Suggestions for further research

At 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 (Jana 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 χ^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* **??** *Kiritchenko 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

[1] *Ricardo Baeza-Yates and Berthier Tibeiro-Neto.* Modern Information Retrieval. *Addison Wesley, 1999.*

[2] *C. Blaschke and A. Valencia. Can bibliograpic pointers for known biological data be found automatically? protein interactions as a case study.* Comparative and Funcional Genomics.

[3] *Nichalas Chase. Understanding sax.* developerWorks. http://www.ibm.com/developerWorks, *2004.*

[4] *Jeff Connor-Linton. Chi square tutorial.* http://www.georgetown.edu/faculty/ballc/webtools/web_chi_tut.html, *2003.*

[5] *The Gene Ontology Consortium. Gene ontology: tool for the unification of biology.* Nature Genetics, *2000.*

[6] *G.Mendel. Experiments in plant hybridization.* http:/www.mendweb.org/Mendel.html, year=1865,.

[7] *G.Saltion and M.J. McGill. Introduction to modern information retrieval.* McGrawHill New York, *1983.*

[8] *Elliotte Rusty Harold.* Processing XML with Java. *Addison-Wesley, 2002.*

[9] *NLP in Bioinformatics.*

[10] *Mitchell JA, Aronson AR, Mork JG, Folk LC, Humphrey SM, and Ward JM. Gene indexing: Chracterization and analysis of nlm's generifs.* Proc AMIA SYMP, *2003.*

[11] *Svetlana Kiritchenko, Stan Matwin, and A.Fazel Famili. Hierarchical text categorization as a tool of associating genes with gene ontology codes.* Proceedings of the Second European Workshop on Data Mining and Text Mining in Bioinformatics.

[12] *Lewis J. Kleinsmith and Valerie M.Kish.* Principles of Cell and Molecular Biology. *HarperCollins College Publishers, 1995.*

[13] *M.F.Porter. An algorithm for suffix stripping.* Program, *1980.*

[14] *A. Moffat and J.Zobel. Self-indexing inverted files for fast text retrieval.* ACM Transactions on Information System, 14(4):349-379, *1996.*

[15] Human Genome Program. http://www.ornl.gov/TechResources/Human_Genome/home.html.

[16] D. Proux, F. Rechenmann, and L. Julliard.

[17] Soumya Raychaudhuri, Jeffrey T. Chang, Patrick D. Sutphin, and Russ B. Altman. Associating genes with gene ontology codes using a mximum entropy analysis of biomedical literature. Genome Res*, 2002.*

[18] Laegreid et al Saetre. Semantic annotation of biomedical literature using google. NTNU*, 2005.*

[19] B. Settles. ABNER: an open source tool for automatically tagging genes, proteins, and other entity names in text http://www.cs.wisc.edu/~bsettles/abner/yagi.html. Bioinformatics*, 2005. To Appear.*

[20] Lars H. Strand. Filtering of biological data in xml-format. Norwegian University of Science and Technology*, 2004.*

[21] Lorraine Tanabe and W. John Wilbur. Tagging gene and protein names in full text articles. Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain*, 2002.*

[22] Henrik Tveit. Towards an automated precedure for annotation of gene products, data- and text mining framework using background information. Master's thesis, Norwegian University of Science and Technology, 2003.

[23] W.B.Frakes and R. Baeza-Yates. Information retrieval: Data structures and algorithms. Prentice Hall, Englewood Cliffs, NJ, USA, 1992*, 2004.*

A appendix

A.1 Java Code

Fetch.java

```
1  import java.awt.*;
2  import java.applet.*;
3  import java.io.*;
4  import java.util.*;
5  import java.net.*;
6  import java.io.*;
7  import java.io.*;
8  import org.xml.sax.*;
9  import org.xml.sax.helpers.DefaultHandler;
10 import javax.xml.parsers.SAXParserFactory;
11 import javax.xml.parsers.ParserConfigurationException;
12 import javax.xml.parsers.SAXParser;
13 import java.util.ArrayList;
14
15 public class Fetch extends DefaultHandler{
16   public String aen, ben;
17   public String tekst = "";
18   public int antall= 0;
19   public int flagg = 0;
20
21   public Fetch(String a, String b,String go,boolean stem,int chi){
22     aen= a;
23     ben= b;
24     System.out.println("er her");
25     try{
26       URL url;
27       URLConnection urlConn;
28       DataOutputStream printout;
29       DataInputStream input;
30       String urlen = "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db
=pubmed" +"&" + "retmode=xml" + "&" + "WebEnv=" + aen + "&" + "query_key=" + ben;
31       url = new URL(urlen);
32       urlConn = url.openConnection();
33       urlConn.setDoInput (true);
34       urlConn.setDoOutput (true);
35       urlConn.setUseCaches (false);
36       urlConn.setRequestProperty
37      ("Content-Type", "application/x-www-form-urlencoded");
38       input = new DataInputStream (urlConn.getInputStream ());
39
40
41       SAXParserFactory factory = SAXParserFactory.newInstance();
42       SAXParser saxParser = factory.newSAXParser();
43       saxParser.parse(input, this);
44       input.close();
45       antall = antall/2;
46       Annotate hoved = new Annotate(tekst,antall,go,stem,chi);
47
48     } catch (SAXException t) {
49       System.out.println("Something wrong when parsing");
50       System.exit(0);
51     } catch(IOException g){
52       System.out.println("IO-fault reading when parsing");
```

```
53          System.exit(0);
54        } catch(ParserConfigurationException abb){
55          System.out.println("Something went wrong setting up the parser, make sure
    you have java JDK1.4 or newer");
56          System.exit(0);
57        }
58
59
60
61      }
62
63      public void characters(char[] ch, int start,
64                     int length) throws SAXException {
65        if(flagg ==1){
66          tekst = tekst + new String(ch,start,length);
67        }
68
69      }
70      public void endElement(String namespaceURI, String localName,
71                     String qName) throws SAXException {
72        flagg = 0;
73      }
74
75      public void startElement(String namespaceURI,
76                       String sName, // simple name (localName)
77                       String qName, // qualified name
78                       Attributes attrs)
79        throws SAXException
80      {
81        if(qName.equals("ArticleTitle")|| qName.equals("AbstractText")){
82          flagg = 1;
83          antall++;
84
85        }
86      }
87
88
89
90  }
```

ArticleFetch.java

```
1   import java.awt.*;
2   import java.applet.*;
3   import java.io.*;
4   import java.util.*;
5   import java.util.Calendar;
6   import java.text.DateFormat;
7   import java.net.*;
8   import java.io.*;
9   import java.io.*;
10  import org.xml.sax.*;
11  import org.xml.sax.helpers.DefaultHandler;
12  import javax.xml.parsers.SAXParserFactory;
13  import javax.xml.parsers.ParserConfigurationException;
14  import javax.xml.parsers.SAXParser;
15  import java.util.ArrayList;
16
17  public class ArticleFetch extends DefaultHandler{
18      public String aen, ben;
19     FileOutputStream out; // declare a file output object
20       PrintStream p; // declare a print stream object
```

```
21      String pubmedid;
22      String a,b;
23      public String pubid;
24      String tekst = "";
25      int tall, flagg;
26      int retstart = 0;
27      int retmax = 500;
28      int jada;
29      String alfa;
30      FileInputStream fstream;
31      DataInputStream in;
32
33      public ArticleFetch (String pubid){
34
35           System.out.println("er i test");
36      this.pubid = pubid;
37         try{
38      URL url;
39           URLConnection   urlConn;
40           DataOutputStream      printout;
41           DataInputStream input;
42
43
44           String urlen = "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi
    ?db=pubmed&id="+ pubid + "&retmode=xml";
45
46           url = new URL (urlen);
47           urlConn = url.openConnection();
48           urlConn.setDoInput (true);
49           urlConn.setDoOutput (true);
50           urlConn.setUseCaches (false);
51           urlConn.setRequestProperty
52            ("Content-Type", "application/x-www-form-urlencoded");
53           input = new DataInputStream (urlConn.getInputStream ());
54
55              SAXParserFactory factory = SAXParserFactory.newInstance();
56
57
58                      SAXParser saxParser = factory.newSAXParser();
59                      saxParser.parse(input, this);
60                   input.close();
61                          // Connect print stream to the output stream
62
63             } catch (SAXException t) {
64                      System.out.println("Something wrong in config file");
65                      System.exit(0);
66                 } catch(IOException g){
67                          System.out.println("IO-fault reading from config file,
    make sure it excists and can be read");
68                          System.exit(0);
69                  } catch(ParserConfigurationException abb){
70                          System.out.println("Something went wrong setting up the
    parser, make sure you have java JDK1.4 or newer");
71                          System.exit(0);
72                       }
73
74
75      }
76
77
78  public String get_article(){
79           return tekst;
```

```
80        }
81
82    public void endDocument() throws SAXException{
83
84
85
86        }
87
88        public void characters(char[] ch, int start,
89                    int length) throws SAXException {
90                    if(flagg ==1){
91                      System.out.println("er her ja");
92                      tekst = tekst + " " + new String(ch,start,length);
93                    }
94
95             }
96        public void endElement(String namespaceURI, String localName,
97                    String qName) throws SAXException {
98                      flagg = 0;
99                    }
100
101       public void startElement(String namespaceURI,
102                   String sName, // simple name (localName)
103                   String qName, // qualified name
104                   Attributes attrs)
105     throws SAXException
106   {
107                 if(qName.equals("ArticleTitle")|| qName.equals("AbstractText")){
108                      flagg = 1;
109
110
111                 }
112       }
113
114
115
116
117
118   }
```

Search.java

```
1   import java.awt.*;
2   import java.applet.*;
3   import java.io.*;
4   import java.util.*;
5   import java.net.*;
6   import java.io.*;
7   import java.io.*;
8   import org.xml.sax.*;
9   import org.xml.sax.helpers.DefaultHandler;
10  import javax.xml.parsers.SAXParserFactory;
11  import javax.xml.parsers.ParserConfigurationException;
12  import javax.xml.parsers.SAXParser;
13  import java.util.ArrayList;
14
15  public class Search extends DefaultHandler{
16
17        public int question = 0;
18        public int antall;
19           public String query_key, webenv,search,goid,tekst;
20
```

```
21      public Search(String searchfor){
22           search = searchfor;
23           System.out.println("er i Øsk");
24           try{
25          URL url;
26          URLConnection    urlConn;
27          DataOutputStream      printout;
28          DataInputStream input;
29          url = new URL ("http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.
fcgi?db=pubmed&term=" + search + "&retmax=500&usehistory=y");
30          urlConn = url.openConnection();
31          urlConn.setDoInput (true);
32          urlConn.setDoOutput (true);
33          urlConn.setUseCaches (false);
34          urlConn.setRequestProperty
35           ("Content-Type", "application/x-www-form-urlencoded");
36          input = new DataInputStream (urlConn.getInputStream ());
37
38           SAXParserFactory factory = SAXParserFactory.newInstance();
39
40
41                    SAXParser saxParser = factory.newSAXParser();
42                    saxParser.parse(input, this);
43                input.close();
44
45
46                 } catch (SAXException t) {
47                   System.out.println("Something wrong in config file");
48                   System.exit(0);
49               } catch(IOException g){
50                        System.out.println("IO-fault reading from config file,
make sure it excists and can be read");
51                        System.exit(0);
52                } catch(ParserConfigurationException a){
53                        System.out.println("Something went wrong setting up the
parser, make sure you have java JDK1.4 or newer");
54                        System.exit(0);
55                    }
56
57                    Fetch get = new Fetch(webenv, query_key);
58                     tekst = get.get_text();
59                     antall = get.get_count();
60
61      }
62
63
64
65  public String get_text(){
66      return tekst;
67      }
68
69  public int get_count(){
70      return antall;
71  }
72
73
74
75      public void characters(char[] ch, int start,
76                   int length) throws SAXException {
77
78                   if(question==1){
```

```
79                      String nystreng = new String(ch,start,length);
80                      webenv = nystreng;
81                        System.out.println(nystreng);
82                        }
83                      if(question==2){
84                      String nystreng= new String(ch, start, length);
85                        query_key = nystreng;
86                        System.out.println(nystreng);
87
88                              }
89              }
90        public void endElement(String namespaceURI, String localName,
91                      String qName) throws SAXException {
92                        question = 0;
93                      }
94
95        public void startElement(String namespaceURI,
96                      String sName, // simple name (localName)
97                      String qName, // qualified name
98                      Attributes attrs)
99      throws SAXException
100     {
101             //System.out.println(qName);
102             if(qName.equals("WebEnv")){
103                   question = 1;
104                   }
105             if(qName.equals("QueryKey")){
106                   System.out.println("jada");
107                         question=2;
108                   }
109
110       }
111
112       }           // End of class Happy.
```

Jdbc.java

```
1   import java.sql.*;
2   import java.util.ArrayList;
3
4   public class Jdbc{
5   public ArrayList array;
6   public Statement stmt;
7   public Connection con;
8   public String buffer="";
9   public int tt = 0;
10        public Jdbc(){
11        connect_database();
12        }
13
14
15        public void connect_database(){
16              try {
17
18         Class.forName("com.mysql.jdbc.Driver");
19         String url =
20               "jdbc:mysql://localhost:3306/sigurd";
21         con =  DriverManager.getConnection(url,"root", "");
22         stmt = con.createStatement();
23
24
25        }catch( Exception e ) {
```

```
26          e.printStackTrace();
27        }//end catch
28         }
29
30
31
32         public ArrayList get_child(String go){
33         try{
34       //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'");
35
36         ResultSet rs = stmt.executeQuery("SELECT child.* FROM term AS parent, term2term,term AS child WHERE parent.acc = "  + "'" + go + "'" + "AND parent.id = term2term.term1_id AND child.id  = term2term.term2_id");
37         array = new ArrayList();
38         while(rs.next()){
39         String hallo = rs.getString(4);
40
41              array.add(hallo);
42
43   }
44        rs.close();
45
46
47
48        }catch( Exception e ) {
49          e.printStackTrace();
50        }//end catch
51    return array;
52    }//end main
53
54   /*
55   public ArrayList getter2(String go){
56         try{
57
58
59         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 + "'");
60
61
62         array = new ArrayList();
63         while(rs.next()){
64         String hallo = rs.getString(4);
65
66              array.add(hallo);
67
68   }
69        rs.close();
70
71
72
73        }catch( Exception e ) {
74          e.printStackTrace();
75        }//end catch
76    return array;
77    }//end main
78
79
80
81   */
```

```
82
83  public void test(String node){
84
85      try{
86          ResultSet rs = stmt.executeQuery("SELECT * FROM term WHERE acc= '" +
    node + "'");
87
88      while(rs.next()){
89      String hallo = rs.getString(3);
90      String jada = rs.getString(4);
91
92      if(hallo.equals("cellular_component")){
93           System.out.println(jada + "tamtam" + hallo);
94           System.exit(-1);
95      }
96
97      }
98      rs.close();
99      }catch( Exception e ) {
100      e.printStackTrace();
101     }//end catch
102
103
104 }
105
106
107
108 public ArrayList ancestor(String go){
109     try{
110
111
112     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+"'");
113
114     array = new ArrayList();
115     while(rs.next()){
116     String hallo = rs.getString(4);
117
118     array.add(hallo);
119
120
121
122 }
123     rs.close();
124
125
126     }catch( Exception e ) {
127       e.printStackTrace();
128     }//end catch
129  return array;
130  }//end main
131
132 public ArrayList get_rchild(String go){
133     try{
134     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");
135
136
137     array = new ArrayList();
```

```
138         while(rs.next()){
139         String hallo = rs.getString(4);
140
141                 array.add(hallo);
142
143 }
144     rs.close();
145
146       //con.close();
147
148     }catch( Exception e ) {
149       e.printStackTrace();
150     }//end catch
151  return array;
152  }//end main
153
154
155
156 }//end class Jdbc
```

Chi.java

```
1  import java.awt.*;
2  import java.applet.*;
3  import java.io.*;
4  import java.util.*;
5  import java.util.List;
6  import java.net.*;
7  import java.lang.*;
8  import java.io.*;
9  import java.io.*;
10 import java.util.regex.Matcher;
11 import org.xml.sax.*;
12 import org.xml.sax.helpers.DefaultHandler;
13 import javax.xml.parsers.SAXParserFactory;
14 import javax.xml.parsers.ParserConfigurationException;
15 import javax.xml.parsers.SAXParser;
16 import java.util.ArrayList;
17 import java.util.regex.Pattern;
18 import java.util.regex.Matcher;
19 import java.util.Calendar;
20 import java.text.DateFormat;
21
22 public class Chi extends DefaultHandler{
23     public int question = 0;
24     public String nystreng,h,aaa,a,beta,ceta,ii,gg,qqq;
25     public String tidligere = "";
26     public BufferedWriter out;
27       public HashMap hashMap,hoho,haaa,hiha;
28       public int jalla2 = 0;
29       public int jalla,question2;
30      public int ja=0;
31      public int hh = 0;
32      public StringBuffer buffer;
33      public int u = 0;
34      public int i = 0;
35      public int flagg = 0;
36      public int[] la,test,antallet;
37      public String[] oo,arr;
38      public HashMap[] hasj,hasj2,chihasj;
39      public HashMap a1,a2,a3;
40      public int jaba,jo,tall;
```

```
41      public int hi = 0;
42      public double total = 0.0;
43      public String str,s4,s1,s2, nystreng_old;
44      public Pattern p;
45      public String[] chiwords;
46      public Matcher m;
47      public int size,totalen;
48      public double g;
49      public int ass=0;
50      public FileInputStream fstream,fstream2,fstreamen;
51      public TreeMap treeMap;
52      public HashMap t;
53      public BufferedReader inja;
54      public DataInputStream inen,in,ina;
55      public Collection[] colla;
56      public Weka weka;
57      public HashMap trening, test2,midler;
58      public Stem stem;
59      public Jdbc jdbc;
60      public int chi;
61      public int index = 0;
62      public boolean stemmen;
63      public ArrayList hasjArray,rowsum, fifi ,fofo ,pubider,goer,klasser;
64      public Chi(ArrayList klasse, Stem stemer,HashMap trene, HashMap test2,
    ArrayList pubide,ArrayList goe,Jdbc jd,boolean stemn,int chinr){
65          chi = chinr;
66          klasser = klasse;
67          stemmen = stemn;
68          jdbc = jd;
69          trening = trene;
70          this.test2 = test2;
71          pubider = pubide;
72          goer = goe;
73          stem = stemer;
74          size = klasse.size();
75          midler = new HashMap();
76          try{
77
78              antallet = new int[klasse.size()];
79              hasjArray = new ArrayList();
80              fifi = new ArrayList();
81              hashMap = new HashMap();
82           treeMap = new TreeMap(hashMap);
83              rowsum = new ArrayList();
84          hasj = new HashMap[size + 2 ];
85              test  = new int[2];
86          for(int a= 0; a < size + 2 ; a++){
87
88              hasjArray.add(new HashMap());
89
90          }
91
92
93
94          for(int a = 0; a < size; a++){
95          hasj[a] = new HashMap(trening.size());
96          }
97
98              Set setting2 = trening.keySet();
99          Iterator iter2 = setting2.iterator();
100
101
```

```
102             while(iter2.hasNext()){
103
104             Object o = iter2.next();
105             s4 = o.toString();
106
107
108                     for(int a = 0; a < size; a++){
109                         hasj[a].put(s4,"0");
110
111                         }
112
113                     for(int i = 0; i < size + 2;i++){
114                     HashMap hasjsj = (HashMap) hasjArray.get(i);
115                     hasjsj.put(s4,"0");
116                     hasjArray.set(i, hasjsj);
117
118                 }
119
120                 }
121
122
123
124
125
126
127             SAXParserFactory factory = SAXParserFactory.newInstance();
128
129
130                     DateFormat df = DateFormat.getTimeInstance();
131                 int tid = df.getCalendar().get(Calendar.HOUR_OF_DAY);
132
133
134                 int tid2 = df.getCalendar().get(Calendar.MINUTE);
135                 int tid3 = df.getCalendar().get(Calendar.SECOND);
136                 System.out.println("START" + "HOUR:" + tid +"MINUTE:" + tid2 + "
    SECOND:" + tid3);
137
138                      SAXParser saxParser = factory.newSAXParser();
139                      saxParser.parse(new File("ja.txt"), this);
140
141
142
143                   } catch (SAXException t) {
144                     System.out.println("Something wrong in config file" + t.
    toString());
145
146                 } catch(IOException g){
147                         System.out.println("IO-fault reading from config file,
    make sure it excists and can be read");
148                         System.exit(0);
149                  } catch(ParserConfigurationException a){
150
151
152                         System.out.println("Something went wrong setting up the
    parser, make sure you have java JDK1.4 or newer");
153                         System.exit(0);
154                     }
155
156             calculate_chi_words();
157             classify();
158         }
159
```

```
160
161
162
163     public void characters(char[] ch, int start,
164                 int length) throws SAXException {
165
166
167                     if(question==3){
168
169                      jaba = Integer.parseInt(new String(ch,start,length));
170
171
172
173
174
175
176                     for(int a = 0; a <pubider.size();a++){
177                           question2=0;
178                           question=0;
179                           int dd = ((Integer)pubider.get(a)).intValue();
180
181                           if(jaba==dd){
182
183
184                                 question2 = 1;
185                                 question=0;
186                                 break;
187                           }
188                           else{
189                                 question2 = 0;
190                                 question = 0;
191                           }
192                     }
193                     }
194
195               if (question==2 && question2==1){
196                     buffer = new StringBuffer();
197                           nystreng = String.valueOf(ch,start,length);
198                           buffer.append(nystreng);
199
200                     buffer.append(" ");
201
202                      nystreng="";
203
204               }
205
206
207                                 if(question==1 && question2==1){
208                                 //System.out.println(jaba);
209                                 buffer.append(String.valueOf(ch, start, length));
210                                 flagg =1;
211                                 nystreng = "";
212                                 }
213
214
215
216          }
217
218     public void endElement(String namespaceURI, String localName,
219                 String qName) throws SAXException {
220
221               if(flagg ==1){
```

```
222                 a = (String)goer.get(index);
223                 index++;
224                         nystreng = buffer.toString();
225                         nystreng_old = nystreng;
226                         Pattern p = Pattern.compile("^(.*)\\s(.*)$");
227                     Matcher m = p.matcher(a);
228                     if(m.matches()){
229                      s1 = m.group(1);
230                      s2 = m.group(2);
231                     }
232
233
234                      jalla = Integer.parseInt(s1);
235
236                     ceta = s2;
237                     //System.out.println("jalla er " + jalla + "jalla2 er " +
    jalla2);
238
239                     ArrayList akk = jdbc.ancestor(s2);
240                     //ArrayList akk = new ArrayList();
241
242                     //counter(nystreng,ceta);
243
244                             for(int i = 0; i < akk.size(); i++){
245
246                                 for(int a = 0; a < klasser.size(); a++){
247
248                                     String ai = (String)akk.get(i);
249                                 String oi = (String)klasser.get(a);
250
251                                     if(ai.equals(oi)){
252
253                                     //System.out.println(nystreng);
254                                     counter(nystreng,(String)akk.get(i));
255                                 }
256                             }
257                         }
258
259
260
261                             buffer = null;
262                             }
263
264
265
266
267                 flagg = 0;
268                 question = 0;
269
270
271                 }
272
273
274     public void startDocument() throws SAXException{
275
276
277     }
278
279
280     public void endDocument() throws SAXException{
281     DateFormat df = DateFormat.getTimeInstance();
282                 int tid = df.getCalendar().get(Calendar.HOUR_OF_DAY);
```

```
283                 int tid2 = df.getCalendar().get(Calendar.MINUTE);
284                 int tid3 = df.getCalendar().get(Calendar.SECOND);
285                 System.out.println("END" + "HOUR:" + tid +"MINUTE:" + tid2 + "
    SECOND:" + tid3);
286                 String q = "";
287                 String s = "";
288           System.out.println(size);
289
290
291
292
293
294
295
296
297
298
299 }
300       public String[] getwords(){
301           System.out.println("lengden er ån " + chiwords.length);
302           return chiwords;
303           }
304       public ArrayList getklasse(){
305           return hasjArray;
306           }
307       public ArrayList getmange(){
308           return klasser;
309           }
310
311
312
313       public void startElement(String namespaceURI,
314                   String sName, // simple name (localName)
315                   String qName, // qualified name
316                   Attributes attrs)
317       throws SAXException
318     {
319
320           if(qName.equals("PMID")){
321                 question = 3;
322                 }
323
324           if(qName.equals("AbstractText")){
325                 question = 1;
326                 }
327           if(qName.equals("ArticleTitle")){
328
329                       question=2;
330                 }
331                 }
332
333
334       public void counter(String streng,String alla){
335
336       //System.out.println(streng +  " "  + alla + "%%%" + hasj[0].size());
337       tall = 0;
338       for(int g = 0; g < klasser.size(); g++){
339           if(((String)klasser.get(g)).equals(alla)){
340                 tall = g;
341                 break;
342           }
343       }
```

```
344         //System.out.println("tallet er " + tall);
345
346         p = Pattern.compile("\\p{Punct}" );
347         m = p.matcher(streng);
348         String juha = m.replaceAll("");
349
350
351         Pattern splitter = Pattern.compile ("[ ]++" );
352         //Pattern splitter = Pattern.compile ("[,.;: ]++" );
353         String[] words = splitter.split(juha);
354         if(stemmen){
355         for(int i = 0; i < words.length; i++){
356               words[i] = stem.hei(words[i]);
357
358         }
359         }
360         else{
361               for(int i = 0; i < words.length; i++){
362               words[i] = words[i].toLowerCase();
363
364               }
365         }
366
367         for(int y = 0; y < words.length; y++){
368
369                     if(hasj[tall].containsKey(words[y])){
370                           midler.put(words[y], "0");
371                           //System.out.println("ggggggggggggggggg");
372                     }
373         }
374
375
376
377
378
379
380                     Set setting1 = midler.keySet();
381                     Iterator iter1 = setting1.iterator();
382
383                           while(iter1.hasNext()){
384                           Object o = iter1.next();
385
386                           String s2 = o.toString();
387
388                           Object oo = hasj[tall].get(s2);
389                           String s3 = oo.toString();
390                           int c = Integer.parseInt(s3);
391
392
393                           c++;
394                           h = Integer.toString(c);
395
396                           hasj[tall].put(s2,h);
397                           }
398
399
400
401         midler.clear();
402         int a = 0;
403
404
405
```

```
406                 for(int bb = 0; bb < size; bb++){
407                     if(alla.equals((String)klasser.get(bb))){
408                         antallet[bb] = antallet[bb] + 1;
409                         hoho = (HashMap) hasjArray.get(bb);
410
411                         a = bb;
412
413                     }
414                 }
415
416
417             for(int i= 0; i < words.length; i++){
418
419                 if(hoho.containsKey(words[i])){
420                 try{
421
422                 Object ai = hoho.get(words[i]);
423
424                 String b = ai.toString();
425
426                 int c = Integer.parseInt(b);
427
428                 //System.out.println(c);
429                 c++;
430
431                 h = Integer.toString(c);
432
433                 }catch(Exception e){
434
435                 }
436
437                 hoho.put(words[i],h);
438
439                 }
440             }
441             //NB kan ævre feil
442             hasjArray.set(a,hoho);
443
444         }
445
446             public void calculate_chi_words(){
447
448             int halla = 0;
449
450
451             for(int i = 0; i < antallet.length; i++){
452             System.out.println(antallet[i]);
453             }
454
455
456
457             for(int i = 0;i < size; i++){
458             System.out.println("####################" + size);
459             double jo = 0.0;
460             Collection collas = ((HashMap) hasjArray.get(i)).values();
461
462             Iterator it = collas.iterator();
463             while(it.hasNext()){
464             Object o = it.next();
465             String ss = o.toString();
466             double c = Double.parseDouble(ss);
467             jo = jo + c;
```

```
468
469
470             }
471             rowsum.add(i,new Double(jo));
472             System.out.println("*****************************************");
473             System.out.println("rowsum" + jo);
474
475
476
477
478             }
479
480                 System.out.println("Østrrelsen av klasser er" + klasser.size());
481             ArrayList aiu = new ArrayList();
482                 for(int b= 0; b < klasser.size(); b++){
483                     if(((Double) rowsum.get(b)).doubleValue()==0.0){
484                         System.out.println("rowsum er 0");
485
486                         aiu.add(new Integer(b));
487                     }
488                 }
489
490                 for(int d=0; d < aiu.size(); d++){
491                     Integer bbb = (Integer) aiu.get(d);
492
493                     System.out.println("rowsum er 3");
494                     int aaa = bbb.intValue();
495                     klasser.remove(aaa-ass);
496                     hasjArray.remove(aaa-ass);
497                     System.out.println("rowsum er 5");
498                     rowsum.remove(aaa-ass);
499                     size--;
500                     ass++;
501                     }
502
503
504                 for(int g=0;g < size; g++){
505                 total = total + ((Double) rowsum.get(g)).doubleValue();
506                 }
507
508                 System.out.println("kom hit %%" + klasser.size() + " "  + rowsum.
    size());
509
510                 // test if rowsum = 0
511                 for(int a = 0; a < rowsum.size(); a++){
512
513                     if(((Double) rowsum.get(a)).doubleValue()==0.0){
514                         System.out.println("Østrrelsen er " +rowsum.size());
515                         System.out.println("rowsum er null");
516                         System.exit(-1);
517                     }
518                 }
519
520
521
522             for(int i = 0; i < size; i++){
523             double c = 0.0;
524             double d = 0.0;
525             a1 = (HashMap) hasjArray.get(i);
526             a2 = (HashMap) hasjArray.get(hasjArray.size()-2);
527             Set setting = a1.keySet();
528             Iterator iter = setting.iterator();
```

```
529
530             while(iter.hasNext()){
531                   Object o = iter.next();
532
533                   String s2 = o.toString();
534
535
536                   String value = (String) a1.get(o);
537
538                   double cc = Double.parseDouble(value);
539
540             String a = (String) a2.get(s2);
541
542             double dd = Double.parseDouble(a);
543
544                   double f = cc+dd;
545
546                   String ii = Double.toString(f);
547
548
549                   a2.put(s2, ii);
550                   hasjArray.set((hasjArray.size() - 2),a2);
551             }
552
553             }
554             System.out.println("kom hit");
555             /*
556             String aaa = "";
557                   a1 = (HashMap) hasjArray.get(hasjArray.size()-2);
558                   Collection collas = a1.values();
559
560                   Iterator it = collas.iterator();
561                   while(it.hasNext()){
562                         Object o = it.next();
563                         String ss = o.toString();
564                         System.out.println(ss);
565                   }
566             */
567
568
569
570
571
572
573             for(int i= 0; i < size ;i++){
574             //14
575
576             a1 = (HashMap) hasjArray.get(i);
577             a2 = (HashMap) hasjArray.get(hasjArray.size()-1);
578             a3 = (HashMap) hasjArray.get(hasjArray.size()-2);
579             Set setting = a1.keySet();
580             Iterator iter = setting.iterator();
581
582             while(iter.hasNext()){
583
584             Object o = iter.next();
585                   String s2 = o.toString();
586
587             String value3 = (String) a1.get(o);
588                   double oo = Double.parseDouble(value3);
589
590                   //System.out.println("er her1 " +oo);
```

```
591                 String columsumed = (String) a3.get(s2);
592
593                 double columsumeds = Double.parseDouble(columsumed);
594
595
596                 //System.out.println("columsumeds " +columsumeds);
597                 double e = ((columsumeds) * ((Double) rowsum.get(i)).doubleValue()
)/total;
598                 //System.out.println("e " +e);
599                 double ny = ((((oo-e) * (oo-e)))/e);
600
601                 /*
602                 if(Double.isNaN(ny)){
603                 System.out.println(s2 + "stoppa her");
604                 }
605                  */
606
607
608                 String valuem = (String) a2.get(o);
609                 double yes = Double.parseDouble(valuem);
610                 double uu = yes + ny;
611
612                 a2.put(s2, Double.toString(uu));
613                 hasjArray.set((hasjArray.size()-1),a2);
614                 //hasj[i].put(s2,Integer.toString(ny));
615
616             }
617
618             }
619
620
621
622
623
624
625             a1 = (HashMap) hasjArray.get(hasjArray.size()-1);
626             Set setting = a1.keySet();
627             Iterator iter = setting.iterator();
628             while(iter.hasNext()){
629                   Object o = iter.next();
630                   String s2 = o.toString();
631
632                   String value3 = (String) a1.get(o);
633                   //System.out.println(s2 + "%%%%%%%%%%" + value3 + " trala");
634
635                   //double oo = Double.parseDouble(value3);
636                   //int aa = (int) oo;
637                   //a1.put(s2, Integer.toString(aa));
638             }
639
640
641
642             System.out.println("er der " +i);
643             int b = 0;
644
645             System.out.println("halla");
646
647             List entrylist = new ArrayList(a1.entrySet());
648             /*
649             for(int a = 0; a < entrylist.size(); a++){
650                       Map.Entry e1 = (Map.Entry) entrylist.get(a);
651                   System.out.println(e1.getKey().toString() + " " + e1.getValue());
```

```
652             }
653             */
654             System.out.println("////////////////////////////");
655
656
657         Collections.sort(entrylist,new ValueComparator());
658             //try{
659             //FileOutputStream out = new FileOutputStream("medstem.txt", true);
660             //PrintStream p = new PrintStream(out);
661             List nyList = entrylist.subList(entrylist.size()-chi, entrylist.size());
662             chiwords = new String[chi];
663             //p.println(nyList.size() + "JADA LISTA ER");
664
665             /*
666             for(int t = 0; t < antallet.length; t++){
667                 p.print(antallet[t] + " ");
668
669                 }
670             p.println();
671             */
672
673
674
675             for(int i = nyList.size() -1; i > -1 ; i--){
676
677                 Map.Entry e1 = (Map.Entry) nyList.get(i);
678                 chiwords[i] = e1.getKey().toString();
679                 System.out.println(e1.getKey().toString() + " " + e1.getValue());
680                 /*
681                 p.print(e1.getKey().toString() + " ");
682                 for(int o = 0; o < size; o++){
683                     Object aff = hasj[o].get(chiwords[i]);
684                     String j = aff.toString();
685                     p.print(j + " ");
686                 }
687             p.println();
688                 */
689         }
690
691
692             System.out.println("closing");
693
694             //out.close();
695
696             System.out.println(nyList.size() + "JADA LISTA ER");
697
698
699             for(int t = 0; t < antallet.length; t++){
700
701                 totalen += antallet[t];
702                 }
703
704
705
706             System.out.println("lengden er ån " + chiwords.length);
707             //}catch(Exception a){System.out.println("noe gikk feil");}
708             //System.out.println("bbbb" + size);
709
710
711
712
713
```

```
714             hasj2 = new HashMap[size];
715             for(int a = 0; a < size; a++){
716             hasj2[a] = new HashMap(chiwords.length);
717             }
718
719
720             /*
721             Set settingq = hasj[1].keySet();
722             Iterator iterq = settingq.iterator();
723
724             while(iterq.hasNext()){
725                   Object o = iterq.next();
726                   String s2 = o.toString();
727                   Object s3 = hasj[2].get(s2);
728
729                   System.out.println(s2 + "&&&&&&&&&&" + s3);
730             }
731
732             */
733       }
734
735
736       public void classify(){
737             System.out.println("Østrrelsen er da "  + size);
738             int qq = 0;
739             for(int a = 0; a < size; a++){
740                   double v = 1.0;
741                   System.out.println("HHHHHHHHHHHHHHHHHHhh");
742                         for(int r = 0; r < chiwords.length; r++){
743
744                               if(test2.containsKey(chiwords[r])){
745                               System.out.println(chiwords[r]);
746                               qq++;
747                               Object aff = hasj[a].get(chiwords[r]);
748                               String j = aff.toString();
749                               System.out.println(j);
750                               double g = Integer.parseInt(j);
751
752                                v = ((1.0 + g))*1000/ 1000*antallet[a];
753       System.out.println((String)klasser.get(a) + "viktigæææææææææææææ " + g + "øø
   " + antallet[a] +"%%% " + v);
754
755                               /*
756                               String s4 = (String) a1.get(chiwords[r]);
757                               double oo = Double.parseDouble(s4);
758                               System.out.println(oo);
759                               double k = v * oo;
760                               */
761                               //System.out.println(g + "YYYYYY" + v + chiwords[r] +
   ¤"" + antallet[a]);
762                               hasj2[a].put(chiwords[r], Double.toString(v));
763                         }
764             }
765             }
766
767             System.out.println("q = " + qq);
768             for(int g = 0; g < size; g++){
769                   double y=1.0;
770                   if(g==0){
771                         System.out.println("Using chiwords:");
772                   }
773                   for(int h = 0; h < chiwords.length; h++){
```

```
774                             if(test2.containsKey(chiwords[h])){
775                             if(g==0){
776
777                             System.out.print(chiwords[h] + " ");
778                             }
779
780
781                             y *= Double.parseDouble(hasj2[g].get(chiwords[h]).
    toString());
782
783                             }
784                     }
785                             double k = y * (double) antallet[g];
786                         System.out.println((String)klasser.get(g) + " "  +k/totalen +
     "## " +antallet[g]);
787             //System.out.println((String)klasser.get(g) + " "  + y);
788             }
789
790
791
792
793
794
795
796             System.out.println("done");
797
798             }
799
800
801
802
803         }           // End of class Happy.
804
805     class ValueComparator implements Comparator {
806       public int compare(Object o1, Object o2) {
807         Map.Entry e1 = (Map.Entry) o1;
808         Map.Entry e2 = (Map.Entry) o2;
809         //Comparable c1 = (Comparable)e1.getValue();
810         //Comparable c2 = (Comparable)e2.getValue();
811
812          Object c1 = e1.getValue();
813          Object c2 = e2.getValue();
814          String c11 = c1.toString();
815          String c22 = c2.toString();
816
817          double a1 = Double.parseDouble(c11);
818          double a2 = Double.parseDouble(c22);
819
820
821
822          return Double.compare(a1,a2);
823          //System.out.println(c1 + "&&&&&&&&&&&&&&&&&&&&&" + c2);
824          //return c1.compareTo(c2);
825     }
826     }
827
```

Annotate.java

```
1   import java.util.regex.Pattern;
2   import java.util.regex.Matcher;
3   import java.util.*;
```

```
4  import java.util.ArrayList;
5  import java.io.*;
6
7
8  public class Annotate{
9
10 public Stem stem;
11 public String s2;
12 public Pattern p;
13 public Pattern splitter;
14 public Matcher m;
15 public int ss = 0;
16 public ArrayList array,ba,goer;
17 public FileInputStream fstream;
18 public DataInputStream in;
19 public ArrayList klasser,ja;
20 public String[] oo;
21 public int hh = 0;
22 public String[] words;
23 public ArrayList a;
24 public int antall,chi_nr;
25 public Jdbc jdbc;
26 public int tell = 0;
27 public String filename,go;
28 public boolean stemn;
29 public String[] hei, stopwords;
30 public HashMap hashMap;
31 public int[] antallet;
32 public ArrayList test;
33 public HashMap mappen;
34 public Annotate(String s,int antall,String go,boolean stemer,int chi_nr){
35     stemn = stemer;
36     this.antall = antall;
37     this.chi_nr = chi_nr;
38     this.go = go;
39     array = new ArrayList();
40     a = new ArrayList();
41     System.out.println("hoved");
42     try {
43     stem = new Stem();
44     antallet = new int[14];
45     hei = tokenstem(s);
46     stopwords = getStopWords();
47     hashMap = new HashMap();
48
49
50     for(int i = 0; i < hei.length; i++){
51         hashMap.put(hei[i],"0");
52     }
53     for(int i = 0; i < hei.length; i++){
54         Object ai = hashMap.get(hei[i]);
55             String b = ai.toString();
56             int c = Integer.parseInt(b);
57             c++;
58             String h = Integer.toString(c);
59
60             hashMap.put(hei[i],h);
61     }
62
63
64     System.out.println("kom hit Øsj");
65     if(go.equals("")){
```

```
66              classify_cache();
67          }
68          else{
69              classify_with_chi();
70          }
71
72      } catch (Exception exception) {
73            exception.printStackTrace();
74        }
75
76      }
77
78      public String[] getStopWords(){
79
80      try{
81      fstream = new FileInputStream("stopwords.txt");
82
83              in = new DataInputStream(fstream);
84               oo = new String[190];
85
86                  while(in.available() !=0){
87                      oo[hh] = in.readLine();
88                      //System.out.println(oo[hh]);
89                      hh++;
90                  }
91              in.close();
92          }catch(Exception h){}
93          return oo;
94
95
96
97      }
98
99
100     public void classify_with_chi(){
101         try{
102                 jdbc = new Jdbc();
103             a = jdbc.get_rchild(go);
104             klasser = jdbc.get_child(go);
105             System.out.println(a.size());
106
107         Set settingr = hashMap.keySet();
108             Iterator iterr = settingr.iterator();
109
110             while(iterr.hasNext()){
111
112             Object o = iterr.next();
113             String s2 = o.toString();
114             String value3 = (String) hashMap.get(o);
115             System.out.println(s2 + " " + value3);
116             }
117
118
119             System.out.println("starting preparsing");
120                Preparse soss = new Preparse(a,jdbc,klasser);
121             System.out.println("done preparsing" );
122             ba = soss.get_text();
123
124
125             array = soss.get_pubmed_id();
126
127
```

```
128
129             goer = soss.get_go();
130
131
132
133
134             test = new ArrayList();
135             mappen = new HashMap();
136             ja = new ArrayList();
137
138             for(int i = 0; i < ba.size(); i++){
139             String[] hallo = tokenstem((String)ba.get(i));
140
141             ja.add(hallo);
142             for(int ii = 0; ii < hallo.length; ii++){
143             mappen.put(hallo[ii],"0");
144             }
145
146             }
147
148             word_counter();
149
150             remove_stopwords();
151
152
153
154             Chicha chi = new Chicha(klasser,stem,mappen, hashMap,array,goer,jdbc,
stemn,chi_nr);
155
156
157         } catch (Exception exception) {
158           exception.printStackTrace();
159       }
160
161
162   }
163
164   public void word_counter(){
165
166         for(int u =0; u < ja.size();u++){
167                 String[] hei2 = (String[])ja.get(u);
168             for(int i = 0; i < hei2.length; i++){
169                 Object ai = mappen.get(hei2[i]);
170                 String b = ai.toString();
171                 int c = Integer.parseInt(b);
172
173                 //System.out.println(c);
174                 c++;
175                 String h = Integer.toString(c);
176
177                 mappen.put(hei2[i],h);
178             }
179             }
180   }
181
182   public void remove_stopwords(){
183
184             Set setting = mappen.keySet();
185             Iterator iter = setting.iterator();
186
187
188             while(iter.hasNext()){
```

```
189
190             Object o = iter.next();
191             s2 = o.toString();
192             String value3 = (String) mappen.get(o);
193             int hg = Integer.parseInt(value3);
194             //if(hg==1){
195
196             //test.add(s2);
197             //}
198
199
200             ss++;
201             }
202
203             for(int i = 0; i < stopwords.length;i++){
204                       //System.out.println(stopwords[i]);
205
206                       mappen.remove(stopwords[i]);
207
208
209
210             }
211
212             for(int z = 0; z <test.size(); z++){
213                   String gg = (String) test.get(z);
214                   mappen.remove(gg);
215             }
216
217 }
218
219 public void classify_cache(){
220       try{
221       System.out.println("inne i cache");
222             Jdbc jdbc = new Jdbc();
223             //a = jdbc.get_rchild("GO:0003674");
224             klasser = jdbc.get_child("GO:0003674");
225       if(stemn){
226        filename = "nystem.txt";
227       }
228       else{
229       filename = "utenstem.txt";
230       }
231       FileInputStream fstream= new FileInputStream(filename);
232             DataInputStream in = new DataInputStream(fstream);
233       String[] chiwords = new String[chi_nr];
234
235       StringTokenizer st = new StringTokenizer("this is a test");
236       HashMap[] hasj = new HashMap[14];
237
238       if(tell==0){
239             int b = 0;
240             String tu = in.readLine();
241             StringTokenizer str = new StringTokenizer(tu);
242
243                   while (str.hasMoreTokens()) {
244                   int d = Integer.parseInt(str.nextToken());
245                   System.out.println(d);
246                   antallet[b] = d;
247                   b++;
248                   }
249       }
250
```

```
251
252         for(int a = 0; a < 14; a++){
253               hasj[a] = new HashMap(chi_nr);
254               }
255
256         while(in.available() !=0 && tell < chi_nr)
257                         {
258
259               String tt = in.readLine();
260               StringTokenizer sts = new StringTokenizer(tt);
261                while (sts.hasMoreTokens()) {
262                     chiwords[tell] = sts.nextToken();
263               for(int a = 0; a < 14; a++){
264                     hasj[a].put(chiwords[tell], sts.nextToken());
265               }
266           }
267         tell++;
268
269
270         }
271         in.close();
272         int size = 14;
273         HashMap[] hasj2 = new HashMap[size];
274               for(int a = 0; a < size; a++){
275               hasj2[a] = new HashMap(chiwords.length);
276               }
277
278
279
280
281
282               System.out.println("Østrrelsen er da "  + size);
283               int qq = 0;
284               double y= 1.0;
285               for(int a = 0; a < size; a++){
286                     double v = 1.0;
287                     y = 1.0;
288
289                           for(int r = 0; r < chiwords.length; r++){
290
291                                 if(hashMap.containsKey(chiwords[r])){
292                                 //System.out.println(chiwords[r]);
293                                 qq++;
294                                 Object aff = hasj[a].get(chiwords[r]);
295                                 String j = aff.toString();
296                                 //System.out.println(j);
297                                 double g = Integer.parseInt(j);
298
299                                  v = ((1.0 + g))*100/ 1000+antallet[a];
300                                  y *= v;
301
302
303                                 //System.out.println(g + "YYYYYY" + v + chiwords[r] +
    ¤"" + antallet[a]);
304                                 //hasj2[a].put(chiwords[r], Double.toString(v));
305                           }
306                     }
307
308                           double k = y * (double) antallet[a];
309                     System.out.println((String)klasser.get(a) + " "  +k/26000.0 +
     " " +antallet[a]);
```

```
310             }
311             /*
312             System.out.println("q = " + qq);
313             for(int g = 0; g < size; g++){
314                   double y=1.0;
315                   if(g==0){
316                         System.out.println("antall documents from medline: " + antall
    );
317                         System.out.println("Using chiwords:");
318                   }
319                   for(int h = 0; h < chiwords.length; h++){
320                               if(hashMap.containsKey(chiwords[h])){
321                               if(g==0){
322
323                               System.out.print(chiwords[h] + " ");
324                               }
325
326                               y *= Double.parseDouble(hasj2[g].get(chiwords[h]).
    toString());
327
328                               }
329                   }
330                               double k = y * (double) antallet[g];
331                         System.out.println((String)klasser.get(g) + " "  +k/26000.0 +
     "## " +antallet[g]);
332             //System.out.println((String)klasser.get(g) + " "  + y);
333             }
334             */
335       } catch (Exception exception) {
336         exception.printStackTrace();
337       }
338
339
340       }
341
342
343
344   public String[] tokenstem(String streng){
345
346       p = Pattern.compile("\\p{Punct}");
347       m = p.matcher(streng);
348       String juha = m.replaceAll("");
349       //Pattern splitter = Pattern.compile ("[,.;: ]++");
350       Pattern splitter = Pattern.compile ("[ ]++");
351
352       String[] words = splitter.split(juha);
353
354       if(stemn){
355             for(int i = 0; i < words.length; i++){
356                   words[i] = stem.hei(words[i]);
357
358             }
359       }
360       else{
361             for(int i = 0; i < words.length; i++){
362                   words[i] = words[i].toLowerCase();
363             }
364
365       }
366
367       return words;
368   }
```

```
369
370
371
372     public static void main(String args[]){
373               boolean stem = false;
374               if(args.length==4){
375                    int tall = Integer.parseInt(args[0]);
376                    if(tall==1){
377                         stem = true;
378                    }
379                    String obtion = args[1];
380                    if(obtion.equals("-g")){
381                         String searchfor = args[2];
382                         int chinumber = Integer.parseInt(args[3]);
383                         //String searchfor = "peg3 AND Parternally expressed
gene";
384                         System.out.println("er her");
385                         //Search a = new Search (searchfor,"",stem,chinumber);
386                         Search a = new Search(searchfor);
387                         String tekst = a.get_text();
388                         int antall = a.get_count();
389                         Annotate anno = new Annotate(tekst,antall, "",stem,
chinumber);
390                         }
391
392                    if(obtion.equals("-t")){
393                         String pubid = args[2];
394                         int chinumber = Integer.parseInt(args[3]);
395                         ArticleFetch fetch = new ArticleFetch(pubid);
396                         String article = fetch.get_article();
397                         //ArticleFetch test = new ArticleFetch(article);
398                         Annotate anno = new Annotate(article,1,"",stem,chinumber
);
399                    }
400                    else{
401                         System.out.println("wrong options, try -t og -g");
402                         System.exit(-1);
403
404                    }
405
406               }
407
408
409               if(args.length==5){
410               int tall = Integer.parseInt(args[0]);
411                    if(tall==1){
412                         stem = true;
413                    }
414               String obtion = args[1];
415                    if(obtion.equals("-g")){
416                         String searchfor = args[2];
417                         int chinumber = Integer.parseInt(args[3]);
418                         String go = args[4];
419                         //String searchfor = "peg3 AND Parternally expressed
gene";
420                         //Search a = new Search (searchfor,go,stem,chinumber);
421                         Search a = new Search(searchfor);
422                         String tekst = a.get_text();
423                         int antall = a.get_count();
424                         Annotate anno = new Annotate(tekst,antall, go,stem,
chinumber);
```

```
425                         }
426
427                     if(obtion.equals("-t")){
428                         String pubid = args[2];
429                         int chinumber = Integer.parseInt(args[3]);
430                         String go = args[4];
431                         ArticleFetch fetch = new ArticleFetch(pubid);
432                         String article = fetch.get_article();
433                         Annotate anno = new Annotate(article,1,go,stem,chinumber
);
434                     }
435                     else{
436                         System.out.println("wrong options, try -t og -g");
437                         System.exit(-1);
438
439                     }
440
441
442
443                 }
444                 else{
445                     System.out.println("Wrong numbers of arguments, must be at
least 4");
446                     System.exit(-1);
447                 }
448
449             }
450
451
452
453 }
```

Test.java

```
1  import java.awt.*;
2  import java.io.*;
3  import java.util.*;
4  import java.util.Calendar;
5  import java.text.DateFormat;
6  import java.net.*;
7  import java.io.*;
8  import org.xml.sax.*;
9  import org.xml.sax.helpers.DefaultHandler;
10 import javax.xml.parsers.SAXParserFactory;
11 import javax.xml.parsers.ParserConfigurationException;
12 import javax.xml.parsers.SAXParser;
13 import java.util.ArrayList;
14
15 public class Download extends DefaultHandler{
16     public String aen, ben;
17    FileOutputStream out; // declare a file output object
18       PrintStream p; // declare a print stream object
19       String pubmedid;
20       String a,b;
21       int tall;
22       int retstart = 0;
23       int retmax = 500;
24       int jada;
25       String alfa;
26       FileInputStream fstream;
27       DataInputStream in;
28
```

```
29      public Download    (){
30           try{
31           fstream = new FileInputStream("avis3.txt");
32           in = new DataInputStream(fstream);
33           URL url;
34           URLConnection  urlConn;
35           DataOutputStream printout;
36           DataInputStream input;
37              alfa = pubmedid;
38
39           while(in.available() !=0){
40                b = "";
41                        for(int i = 0; i < 500; i++){
42                if(in.available() !=0){
43                     a = in.readLine() + ",";
44
45                 b = b + a;
46                }
47                }
48
49           pubmedid = b.substring(0, b.length() -1 );
50           if(pubmedid.equals("")){
51                System.out.println("er her");
52                break;
53                }
54
55
56           //System.out.println(pubmedid);
57           System.out.println("retstart er" + retstart);
58
59
60           String urlen = "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi
   ?db=pubmed&id="+ pubmedid +"&retmax=" + retmax + "&retmode=xml";
61
62           url = new URL (urlen);
63           urlConn = url.openConnection();
64           urlConn.setDoInput (true);
65           urlConn.setDoOutput (true);
66           urlConn.setUseCaches (false);
67           urlConn.setRequestProperty
68            ("Content-Type", "application/x-www-form-urlencoded");
69           input = new DataInputStream (urlConn.getInputStream ());
70
71             out = new FileOutputStream("myfile.xml",true);
72
73                        // Connect print stream to the output stream
74         p = new PrintStream( out );
75
76
77           System.out.println("hei");
78
79             String str;
80           while (null != ((str = input.readLine())))
81            {
82            p.println(str);
83            }
84
85                   p.close();
86                       input.close();
87
88                DateFormat df = DateFormat.getTimeInstance();
89                int seconds = df.getCalendar().get(Calendar.SECOND);
```

```
90                  System.out.println(seconds);
91                  tall = df.getCalendar().get(Calendar.SECOND);
92                  while(tall  < (seconds + 2)){
93                  DateFormat d = DateFormat.getTimeInstance();
94                  tall = d.getCalendar().get(Calendar.SECOND);
95                  }
96                  System.out.println(a);
97
98
99                  } //for
100
101
102           } catch (Exception e)
103           {
104           System.err.println("Exception: " + e);
105           }
106
107
108
109
110                 }//end of konstructor
111
112       public static void main(String args[]){
113            Test test = new Test();
114            }
115
116     }           // End of class Happy.
```

Download.java

```
1   import java.awt.*;
2   import java.io.*;
3   import java.util.*;
4   import java.util.Calendar;
5   import java.text.DateFormat;
6   import java.net.*;
7   import java.io.*;
8   import org.xml.sax.*;
9   import org.xml.sax.helpers.DefaultHandler;
10  import javax.xml.parsers.SAXParserFactory;
11  import javax.xml.parsers.ParserConfigurationException;
12  import javax.xml.parsers.SAXParser;
13  import java.util.ArrayList;
14
15  public class Download extends DefaultHandler{
16      public String aen, ben;
17     FileOutputStream out; // declare a file output object
18       PrintStream p; // declare a print stream object
19       String pubmedid;
20       String a,b;
21       int tall;
22       int retstart = 0;
23       int retmax = 500;
24       int jada;
25       String alfa;
26       FileInputStream fstream;
27       DataInputStream in;
28
29       public Download    (){
30            try{
31            fstream = new FileInputStream("avis3.txt");
32            in = new DataInputStream(fstream);
```

```
33            URL url;
34            URLConnection  urlConn;
35            DataOutputStream printout;
36            DataInputStream input;
37               alfa = pubmedid;
38
39            while(in.available() !=0){
40                  b = "";
41                           for(int i = 0; i < 500; i++){
42                  if(in.available() !=0){
43                        a = in.readLine() + ",";
44
45                   b = b + a;
46                  }
47                  }
48
49            pubmedid = b.substring(0, b.length() -1 );
50            if(pubmedid.equals("")){
51                  System.out.println("er her");
52                  break;
53                  }
54
55
56           //System.out.println(pubmedid);
57           System.out.println("retstart er" + retstart);
58
59
60           String urlen = "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi
    ?db=pubmed&id="+ pubmedid +"&retmax=" + retmax + "&retmode=xml";
61
62           url = new URL (urlen);
63           urlConn = url.openConnection();
64           urlConn.setDoInput (true);
65           urlConn.setDoOutput (true);
66           urlConn.setUseCaches (false);
67           urlConn.setRequestProperty
68            ("Content-Type", "application/x-www-form-urlencoded");
69           input = new DataInputStream (urlConn.getInputStream ());
70
71             out = new FileOutputStream("myfile.xml",true);
72
73                           // Connect print stream to the output stream
74          p = new PrintStream( out );
75
76
77           System.out.println("hei");
78
79             String str;
80           while (null != ((str = input.readLine())))
81            {
82            p.println(str);
83            }
84
85                   p.close();
86                      input.close();
87
88                  DateFormat df = DateFormat.getTimeInstance();
89                  int seconds = df.getCalendar().get(Calendar.SECOND);
90                  System.out.println(seconds);
91                  tall = df.getCalendar().get(Calendar.SECOND);
92                  while(tall < (seconds + 2)){
93                  DateFormat d = DateFormat.getTimeInstance();
```

```
94                  tall = d.getCalendar().get(Calendar.SECOND);
95                  }
96                  System.out.println(a);
97
98
99                  } //for
100
101
102           } catch (Exception e)
103           {
104           System.err.println("Exception: " + e);
105           }
106
107
108
109
110                 }//end of konstructor
111
112       public static void main(String args[]){
113            Test test = new Test();
114            }
115
116     }           // End of class Happy.
```

Stem.java

```
1   /*
2
3      Porter stemmer in Java. The original paper is in
4
5          Porter, 1980, An algorithm for suffix stripping, Program, Vol. 14,
6          no. 3, pp 130-137,
7
8      See also http://www.tartarus.org/~martin/PorterStemmer
9
10     History:
11
12     Release 1
13
14     Bug 1 (reported by Gonzalo Parra 16/10/99) fixed as marked below.
15     The words 'aed', 'eed', 'oed' leave k at 'a' for step 3, and b[k-1]
16     is then out outside the bounds of b.
17
18     Release 2
19
20     Similarly,
21
22     Bug 2 (reported by Steve Dyrdahl 22/2/00) fixed as marked below.
23     'ion' by itself leaves j = -1 in the test for 'ion' in step 5, and
24     b[j] is then outside the bounds of b.
25
26     Release 3
27
28     Considerably revised 4/9/00 in the light of many helpful suggestions
29     from Brian Goetz of Quiotix Corporation (brian@quiotix.com).
30
31     Release 4
32
33  */
34
35  import java.io.*;
36
```

```
37  /**
38    * Stemmer, implementing the Porter Stemming Algorithm
39    *
40    * The Stemmer class transforms a word into its root form.  The input
41    * word can be provided a character at time (by calling add()), or at once
42    * by calling one of the various stem(something) methods.
43    */
44
45  public class Stem
46  {  private char[] b;
47     private int i,     /* offset into b */
48                 i_end, /* offset to end of stemmed word */
49                 j, k;
50     private static final int INC = 50;
51                       /* unit of size whereby b is increased */
52     public Stem()
53     {  b = new char[INC];
54        i = 0;
55        i_end = 0;
56     }
57
58     /**
59      * Add a character to the word being stemmed.  When you are finished
60      * adding characters, you can call stem(void) to stem the word.
61      */
62
63     public void add(char ch)
64     {  if (i == b.length)
65        {  char[] new_b = new char[i+INC];
66           for (int c = 0; c < i; c++) new_b[c] = b[c];
67           b = new_b;
68        }
69        b[i++] = ch;
70     }
71
72
73     /** Adds wLen characters to the word being stemmed contained in a portion
74      * of a char[] array. This is like repeated calls of add(char ch), but
75      * faster.
76      */
77
78     public void add(char[] w, int wLen)
79     {  if (i+wLen >= b.length)
80        {  char[] new_b = new char[i+wLen+INC];
81           for (int c = 0; c < i; c++) new_b[c] = b[c];
82           b = new_b;
83        }
84        for (int c = 0; c < wLen; c++) b[i++] = w[c];
85     }
86
87     /**
88      * After a word has been stemmed, it can be retrieved by toString(),
89      * or a reference to the internal buffer can be retrieved by getResultBuffer
90      * and getResultLength (which is generally more efficient.)
91      */
92     public String toString() { return new String(b,0,i_end); }
93
94     /**
95      * Returns the length of the word resulting from the stemming process.
96      */
97     public int getResultLength() { return i_end; }
98
```

```
99      /**
100      * Returns a reference to a character buffer containing the results of
101      * the stemming process.  You also need to consult getResultLength()
102      * to determine the length of the result.
103      */
104     public char[] getResultBuffer() { return b; }
105
106     /* cons(i) is true <=> b[i] is a consonant. */
107
108     private final boolean cons(int i)
109     {  switch (b[i])
110        {  case 'a': case 'e': case 'i': case 'o': case 'u': return false;
111           case 'y': return (i==0) ? true : !cons(i-1);
112           default: return true;
113        }
114     }
115
116     /* m() measures the number of consonant sequences between 0 and j. if c is
117        a consonant sequence and v a vowel sequence, and <..> indicates arbitrary
118        presence,
119
120           <c><v>       gives 0
121           <c>vc<v>     gives 1
122           <c>vcvc<v>   gives 2
123           <c>vcvcvc<v> gives 3
124           ....
125     */
126
127     private final int m()
128     {  int n = 0;
129        int i = 0;
130        while(true)
131        {  if (i > j) return n;
132           if (! cons(i)) break; i++;
133        }
134        i++;
135        while(true)
136        {  while(true)
137           {  if (i > j) return n;
138                 if (cons(i)) break;
139                 i++;
140           }
141           i++;
142           n++;
143           while(true)
144           {  if (i > j) return n;
145              if (! cons(i)) break;
146              i++;
147           }
148           i++;
149         }
150     }
151
152     /* vowelinstem() is true <=> 0,...j contains a vowel */
153
154     private final boolean vowelinstem()
155     {  int i; for (i = 0; i <= j; i++) if (! cons(i)) return true;
156        return false;
157     }
158
159     /* doublec(j) is true <=> j,(j-1) contain a double consonant. */
160
```

```
161     private final boolean doublec(int j)
162     {  if (j < 1) return false;
163        if (b[j] != b[j-1]) return false;
164        return cons(j);
165     }
166
167     /* cvc(i) is true <=> i-2,i-1,i has the form consonant - vowel - consonant
168        and also if the second c is not w,x or y. this is used when trying to
169        restore an e at the end of a short word. e.g.
170
171           cav(e), lov(e), hop(e), crim(e), but
172           snow, box, tray.
173
174     */
175
176     private final boolean cvc(int i)
177     {  if (i < 2 || !cons(i) || cons(i-1) || !cons(i-2)) return false;
178        {  int ch = b[i];
179           if (ch == 'w' || ch == 'x' || ch == 'y') return false;
180        }
181        return true;
182     }
183
184     private final boolean ends(String s)
185     {  int l = s.length();
186        int o = k-l+1;
187        if (o < 0) return false;
188        for (int i = 0; i < l; i++) if (b[o+i] != s.charAt(i)) return false;
189        j = k-l;
190        return true;
191     }
192
193     /* setto(s) sets (j+1),...k to the characters in the string s, readjusting
194        k. */
195
196     private final void setto(String s)
197     {  int l = s.length();
198        int o = j+1;
199        for (int i = 0; i < l; i++) b[o+i] = s.charAt(i);
200        k = j+l;
201     }
202
203     /* r(s) is used further down. */
204
205     private final void r(String s) { if (m() > 0) setto(s); }
206
207     /* step1() gets rid of plurals and -ed or -ing. e.g.
208
209           caresses  ->  caress
210           ponies    ->  poni
211           ties      ->  ti
212           caress    ->  caress
213           cats      ->  cat
214
215           feed      ->  feed
216           agreed    ->  agree
217           disabled  ->  disable
218
219           matting   ->  mat
220           mating    ->  mate
221           meeting   ->  meet
222           milling   ->  mill
```

```
223             messing   -> mess
224
225             meetings  -> meet
226
227     */
228
229     private final void step1()
230     {  if (b[k] == 's')
231        {  if (ends("sses")) k -= 2; else
232           if (ends("ies")) setto("i"); else
233           if (b[k-1] != 's') k--;
234        }
235        if (ends("eed")) { if (m() > 0) k--; } else
236        if ((ends("ed") || ends("ing")) && vowelinstem())
237        {  k = j;
238           if (ends("at")) setto("ate"); else
239           if (ends("bl")) setto("ble"); else
240           if (ends("iz")) setto("ize"); else
241           if (doublec(k))
242           {  k--;
243              {  int ch = b[k];
244                 if (ch == 'l' || ch == 's' || ch == 'z') k++;
245              }
246           }
247           else if (m() == 1 && cvc(k)) setto("e");
248        }
249     }
250
251     /* step2() turns terminal y to i when there is another vowel in the stem. */
252
253     private final void step2() { if (ends("y") && vowelinstem()) b[k] = 'i'; }
254
255     /* step3() maps double suffices to single ones. so -ization ( = -ize plus
256        -ation) maps to -ize etc. note that the string before the suffix must give
257        m() > 0. */
258
259     private final void step3() { if (k == 0) return; /* For Bug 1 */ switch (b[k
    -1])
260     {
261         case 'a': if (ends("ational")) { r("ate"); break; }
262                   if (ends("tional")) { r("tion"); break; }
263                   break;
264         case 'c': if (ends("enci")) { r("ence"); break; }
265                   if (ends("anci")) { r("ance"); break; }
266                   break;
267         case 'e': if (ends("izer")) { r("ize"); break; }
268                   break;
269         case 'l': if (ends("bli")) { r("ble"); break; }
270                   if (ends("alli")) { r("al"); break; }
271                   if (ends("entli")) { r("ent"); break; }
272                   if (ends("eli")) { r("e"); break; }
273                   if (ends("ousli")) { r("ous"); break; }
274                   break;
275         case 'o': if (ends("ization")) { r("ize"); break; }
276                   if (ends("ation")) { r("ate"); break; }
277                   if (ends("ator")) { r("ate"); break; }
278                   break;
279         case 's': if (ends("alism")) { r("al"); break; }
280                   if (ends("iveness")) { r("ive"); break; }
281                   if (ends("fulness")) { r("ful"); break; }
282                   if (ends("ousness")) { r("ous"); break; }
283                   break;
```

```
284         case 't': if (ends("aliti")) { r("al"); break; }
285                   if (ends("iviti")) { r("ive"); break; }
286                   if (ends("biliti")) { r("ble"); break; }
287                   break;
288         case 'g': if (ends("logi")) { r("log"); break; }
289     } }
290
291     /* step4() deals with -ic-, -full, -ness etc. similar strategy to step3. */
292
293     private final void step4() { switch (b[k])
294     {
295         case 'e': if (ends("icate")) { r("ic"); break; }
296                   if (ends("ative")) { r(""); break; }
297                   if (ends("alize")) { r("al"); break; }
298                   break;
299         case 'i': if (ends("iciti")) { r("ic"); break; }
300                   break;
301         case 'l': if (ends("ical")) { r("ic"); break; }
302                   if (ends("ful")) { r(""); break; }
303                   break;
304         case 's': if (ends("ness")) { r(""); break; }
305                   break;
306     } }
307
308     /* step5() takes off -ant, -ence etc., in context <c>vcvc<v>. */
309
310     private final void step5()
311     {   if (k == 0) return; /* for Bug 1 */ switch (b[k-1])
312         {  case 'a': if (ends("al")) break; return;
313            case 'c': if (ends("ance")) break;
314                      if (ends("ence")) break; return;
315            case 'e': if (ends("er")) break; return;
316            case 'i': if (ends("ic")) break; return;
317            case 'l': if (ends("able")) break;
318                      if (ends("ible")) break; return;
319            case 'n': if (ends("ant")) break;
320                      if (ends("ement")) break;
321                      if (ends("ment")) break;
322                      /* element etc. not stripped before the m */
323                      if (ends("ent")) break; return;
324            case 'o': if (ends("ion") && j >= 0 && (b[j] == 's' || b[j] == 't'))
    break;
325                                          /* j >= 0 fixes Bug 2 */
326                      if (ends("ou")) break; return;
327                      /* takes care of -ous */
328            case 's': if (ends("ism")) break; return;
329            case 't': if (ends("ate")) break;
330                      if (ends("iti")) break; return;
331            case 'u': if (ends("ous")) break; return;
332            case 'v': if (ends("ive")) break; return;
333            case 'z': if (ends("ize")) break; return;
334            default: return;
335         }
336         if (m() > 1) k = j;
337     }
338
339     /* step6() removes a final -e if m() > 1. */
340
341     private final void step6()
342     {  j = k;
343        if (b[k] == 'e')
344        {  int a = m();
```

```
345             if (a > 1 || a == 1 && !cvc(k-1)) k--;
346         }
347         if (b[k] == 'l' && doublec(k) && m() > 1) k--;
348     }
349
350     /** Stem the word placed into the Stemmer buffer through calls to add().
351      * Returns true if the stemming process resulted in a word different
352      * from the input.  You can retrieve the result with
353      * getResultLength()/getResultBuffer() or toString().
354      */
355     public void stem()
356     {  k = i - 1;
357         if (k > 1) { step1(); step2(); step3(); step4(); step5(); step6(); }
358         i_end = k+1; i = 0;
359     }
360
361     /** Test program for demonstrating the Stemmer.  It reads text from a
362      * a list of files, stems each word, and writes the result to standard
363      * output. Note that the word stemmed is expected to be in lower case:
364      * forcing lower case must be done outside the Stemmer class.
365      * Usage: Stemmer file-name file-name ...
366      */
367
368
369       public String hei(String hallo){
370
371       String a = hallo.toLowerCase();
372       char[] b = a.toCharArray();
373
374       for(int i = 0; i < b.length; i++){
375             if(!Character.isLetter(b[i])){
376                   return a;
377             }
378       }
379
380       for(int i = 0; i < b.length; i++){
381             if(Character.isLetter(b[i])){
382                   add(b[i]);
383             }
384
385       }
386       stem();
387       String u;
388       u = toString();
389       return u;
390
391       }
392 }
```

B Manual annotations for the test genes

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

C Result output

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

Table 10: java -Xmx500m Annotate 1 -g LGALS3 100

GO-id	Classification probability	Documents in subtree
GO:0000170	2.8913024086448 53E22	2
GO:0003826	1.1046997714076678E35	3
GO:0004033	2.339292758016471E93	15
GO:0004154	128.57399103139014	1
GO:0004497	7.2671352784932 12E180	191
GO:0004800	1.4001102045494227E92	9
GO:0009974	0.8609865470852018	1
GO:0010242	9.035320027015166E20	2
GO:0015002	1.09230316691397 84E125	40
GO:0015036	3.66257680604283 34E88	14
GO:0016614	1.41635305330689 86E182	207
GO:0016627	2.8849852764388115E145	72
GO:0016638	9.10458800684152E129	43
GO:0016645	9.623181100836585E122	35
GO:0016651	5.479699046251034E167	109
GO:0016661	2.2157065137444332E56	5
GO:0016667	3.0141022029931454E133	51
GO:0016675	1.0923031669139784E125	40
GO:0016679	3.5212370504154435E134	30
GO:0016684	4.155643798687955E140	45
GO:0016701	3.62735111279747 54E137	46
GO:0016705	4.284736640419215E184	181
GO:0016721	3.0786350065164925E110	19
GO:0016722	3.099879407369051E85	12
GO:0016725	2.2911885201793725E85	10
GO:0016730	8.209888318431527E42	3
GO:0016903	1.41964172607434 93E163	96
GO:0030611	1.6106665578046756E54	4
GO:0030613	1.0923031669139783E45	4
GO:0051213	6.523889514333432E139	48

Table 11: java -Xmx500m Annotate 1 -g ADH1 400 GO:0016491

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

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