Automatic GO Term Prediction on UniProtKB

Diploma Thesis

at

University of Applied Sciences Weihenstephan

Department of Biotechnology and Bioinformatics

Barbara Meier

23. Mar. 2007

conducted at

European Bioinformatics Institute, Hinxton, Cambridge, UK

Supervised by:

Daniela Wieser
EMBL European Bioinformatics Institute
Wellcome Trust Genome Campus
Hinxton
Cambridge CB10 1SD
United Kingdom

Prof. Dr. Bernhard Haubold
University of Applied Sciences
Weihenstephan
Dept. of Biotechnology & Bioinformatics
85350 Freising
Germany

Erklärung zur Urheberschaft

Gemäß § 31 Abs. 7 der Rahmenprüfungsordnung für die Fachhochschulen (RaPO):

Ich erkläre hiermit, dass die vorliegende Arbeit von mir selbst und ohne fremde Hilfe verfasst und noch nicht anderweitig für Prüfungszwecke vorgelegt wurde. Es wurden keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt. Wörtliche und sinngemäße Zitate sind als solche gekennzeichnet.

Cambridge, den 23/03/2007

_____

Acknowledgements

This work has been supported by a number of people in various ways. My supervisors at the EBI, Dani Wieser and Ernst Kretschmann gave lots of food for thought, helped in many discussions and supported wherever they could. Together with all the other colleagues there, they made the work at the institute very enjoyable. Other support came from Prof. Haubold, my supervisor at the University, reassuring me at a time when I really needed it and giving valuable feedback on the monthly reports and all my questions. I received quite a different but almost even more important help during the time I spent in England from Andi, who accompanied me through all my ups and downs, tried to understand this work even though he didn't have a clue about the subject and helped finding various mistakes in this text. Also from back home of course my family supported me in all circumstances. My father encouraging me and helping through financial support, my mother in listening and all of the family by enjoying a few days of holidays in England. Henning as a team leader and landlord also provided an important part of the basis for this work. Living at his house for most of my time in Cambridge was quite perfect and he made this work possible after all. Also very valuable contribution came from Emily Dimmer and Rebecca Foulger, proofreading this thesis.

Many many thanks!

Contents

1 Summary

Protein annotation in UniProtKB/TrEMBL is poor, however this is the largest and fastest growing part of the Universal Protein Knowledgebase (UniProtKB), and the sequence database group at the EBI aims to automatically enrich the information contained in it. For this purpose knowledge is extracted out of UniProtKB/Swiss-Prot, the manually curated and well annotated part of UniProtKB, and applied to TrEMBL. Over the last eight years, automatic annotation of different protein attributes such as Keywords, comments and features have been successfully implemented.

This thesis investigates machine learning techniques on Gene Ontology (GO) terms in UniProtKB based on the existing methods and also some new approaches. Spearmint, the Decision Tree (DT) algorithm designed by E. Kretschmann [1] for Keyword prediction was tested and found to perform unsatisfactorily on GO term prediction. A first attempt to improve GO term prediction was a simple mapping of Swiss-Prot Keywords to GO terms. The same algorithm was also applied to an InterPro to GO mapping. This mapping, using Bayes's conditional probabilities for rule generation, already lead to a significant improvement in comparison to the Decision Tree approach, but also left room for further refinement. Since manual GO annotation in Swiss-Prot provides a much less complete information base than for Keywords and other attributes, another idea to solving the problem was to enrich the database with additional GO terms without changing the original meaning of the annotation. Due to missing negative associations, no machine learning algorithm can successfully be applied to UniProtKB GO terms. Hence a further step during the enrichment process was to find as reliable as possible GO term exclusions. That way enriched proteins were then again fed into the existing Decision Tree.

Despite some slight improvements in comparison to the original Spearmint run due to the positive enrichment, the results were still not satisfactory. The negative search did not produce a sufficient number of true negative classifications. Hence GO data in Swiss-Prot has turned out to be more difficult to use in automatic annotation than expected.

2 Introduction

2.1 Background

As more and more biological sequence data are produced automatically from high-throughput experiments, the number of entries in databases like the Universal Protein Knowledgebase (UniProtKB) is increasing rapidly. For example the number of entries in UniProtKB has increased by 7% (237,288 new entries) during the short time period between two releases (9.4 on 26th December 2006 and 9.6 on 6th February 2007). Unfortunately most records retrieved from automated processes lack annotations. The fraction of well annotated or manually curated entries in protein or gene databases is decreasing at a speed directly proportional to the increasing number of protein sequence submissions. UniProtKB is a central database containing protein sequences and providing accurate, consistent and rich sequence and functional information where available. It is a union of the Swiss-Prot and TrEMBL datasets and currently contains 3,766,477 protein sequence entries[1]. High quality records are found in the Swiss-Prot part which holds 252,616 (~7% of the

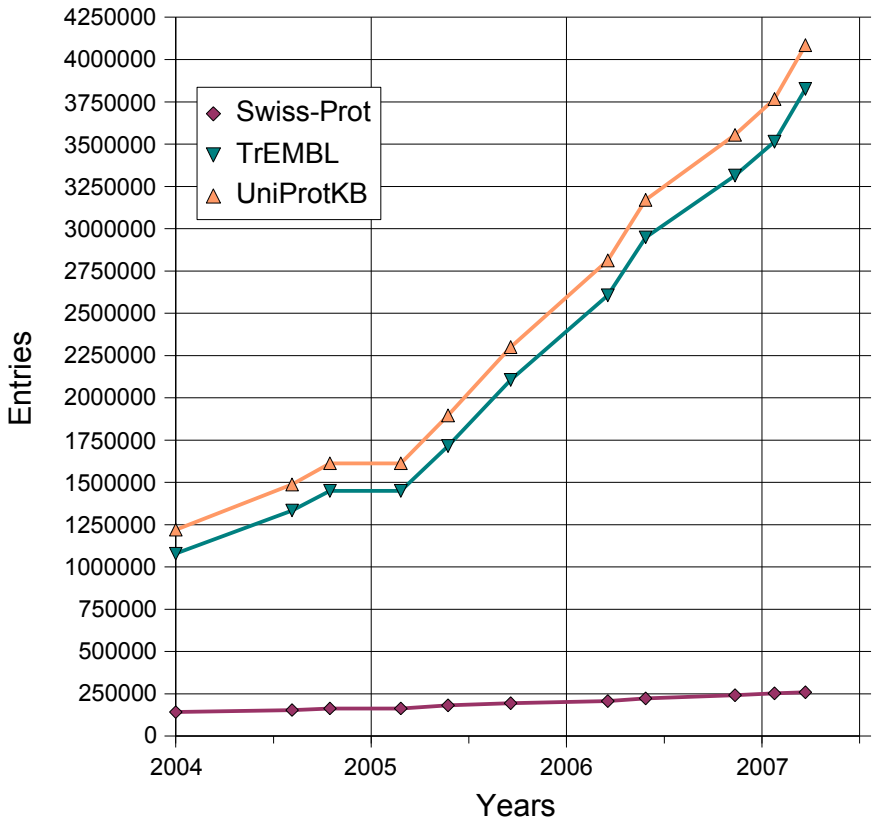

Figure 2.1: Database growth since beginning of 2004. Data out of UniProtKB release notes 1.0 to 9.7

1 Figures for UniProtKB Release 9.4 from 26th of December 2006

UniProtKB total) records with information extracted from literature and curator-evaluated computational analysis. Most of these entries are completely annotated. But the second and far greater part, i.e. UniProtKB/TrEMBL, holds 3,513,861 (~93% of UniProtKB total) records, which are only computationally analysed. While awaiting manual curation and incorporation into UniProtKB/Swiss-Prot, they are enriched with automatic annotation and classification, but annotation is largely incomplete [2, 3, 4, 5]. Figure 2.1 visualises the development of both parts of UniProtKB over the past few years and shows the slow increase in manually curated entries (UniProtKB/Swiss-Prot) and the steep increase of automatically annotated protein entries (UniProtKB/TrEMBL).

UniProtKB provides a broad range of information about proteins. Annotations describe areas such as the function(s) of the protein, post translational modification(s), domains and sites, and secondary structure. An example of a Swiss-Prot entry can be found in Appendix A. Entries are divided into several topics/fields, describing the sequence in general, its organism origin, literature references and function, as well as database cross-references. Functional descriptions in this context are for example Keywords, features, comments and also GO terms:

- Swiss-Prot Keywords (KW) is a controlled vocabulary which is divided into 10 domains. There are 892 distinct Keywords describing the biological functions, cellular components and molecular processes of entries as well as diseases, coding sequence diversities, technical terms, developmental stages, post translational modifications, domains and ligands of proteins.
- The feature table (FT) describes regions or sites of interest in the sequence. In general the feature table lists post translational modifications, binding sites, enzyme active sites, local secondary structure or other characteristics reported in the cited references such as natural variants or isoforms.
- Comments (CC) are largely free text additions to an entry and contain further useful information about a protein. They are arranged into 27 “topics” including function, developmental stage, tissue specificity, similarity and interaction.
- GO terms, the subject of this thesis, are arranged in a controlled vocabulary, similar to Keywords. The 22,929 distinct terms are divided into three domains describing the molecular process(es), cellular component(s) and biological function(s) of gene products. In contrast to Keywords however, GO terms are arranged in a complex hierarchy. In the UniProtKB record they are found as database cross-references. More details about GO terms will be given later in this work.

As mentioned before, Swiss-Prot and TrEMBL differ widely at the annotation level. To show this gap, Figure 2.2 and Table 1 visualise the distribution of the annotations named above. Each column shows the percentage of all Swiss-Prot or TrEMBL entries holding at least one annotation of the named type. In Swiss-Prot, Keywords, features and comments are present in 98% (248,623 entries), 100% (252,616 entries) and 96% (243,681 entries) of all records respectively. In TrEMBL 72% (2,536,466) of the entries have Keywords, whereas only 28% (981,682 entries) show feature tables

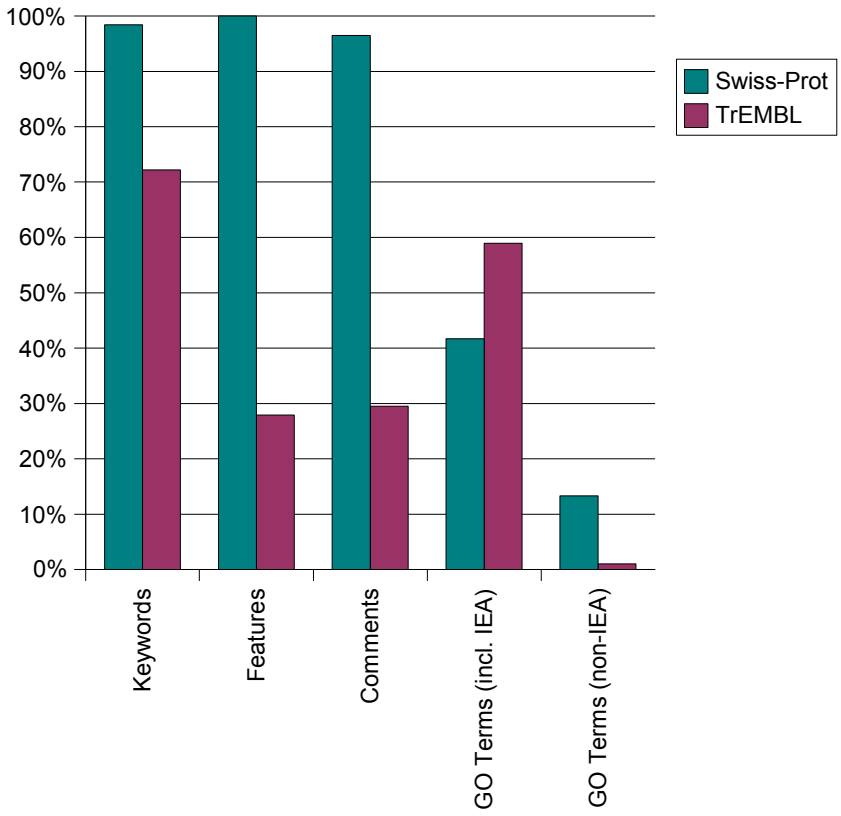

Figure 2.2: Annotated entries in UniProtKB/Swiss-Prot and UniProtKB/TrEMBL as a percentage of all Swiss-Prot or TrEMBL entries respectively. IEA is short for "Inferred from Electronic Annotation".

	Swiss-Prot		TrEMBL	
	absolute	percentage	absolute	percentage
Keywords	248,623	98%	2,536,466	72%
Feature Table	252,616	100%	981,682	28%
Comments	243,681	96%	1,037,020	30%
GO Terms (incl. IEA)	105,301	42%	2,070,472	59%
GO Terms (non-IEA)	33,624	13%	37,260	1%
All Proteins	252,616	100%	3,513,861	100%

Table 1: Distribution of annotated entries in UniProtKB/Swiss-Prot and UniProtKB/TrEMBL as absolute values and percentage of all Swiss-Prot or TrEMBL entries respectively. IEA is short for "Inferred from Electronic Annotation".

and 30% (1,037,020 records) have comments. As per release statistics of TrEMBL and Swiss-Prot, a typical Swiss-Prot entry contains approximately 4 comments and 7 feature annotations whereas an average TrEMBL entry has 0.4 comments and 0.5 feature entries. Separate investigations for Keywords showed an average of 5 per Swiss-Prot entry and 1.5 per TrEMBL entry (see Figure 2.3). Concerning the high coverage of annotations in Swiss-Prot, this part of UniProtKB is predestined to serve as a knowledge base for diverse machine learning algorithms to enrich TrEMBL annotations automatically. Note that the above figures for TrEMBL already include 8 years of automatic annotation work, generating most of the annotation in TrEMBL.

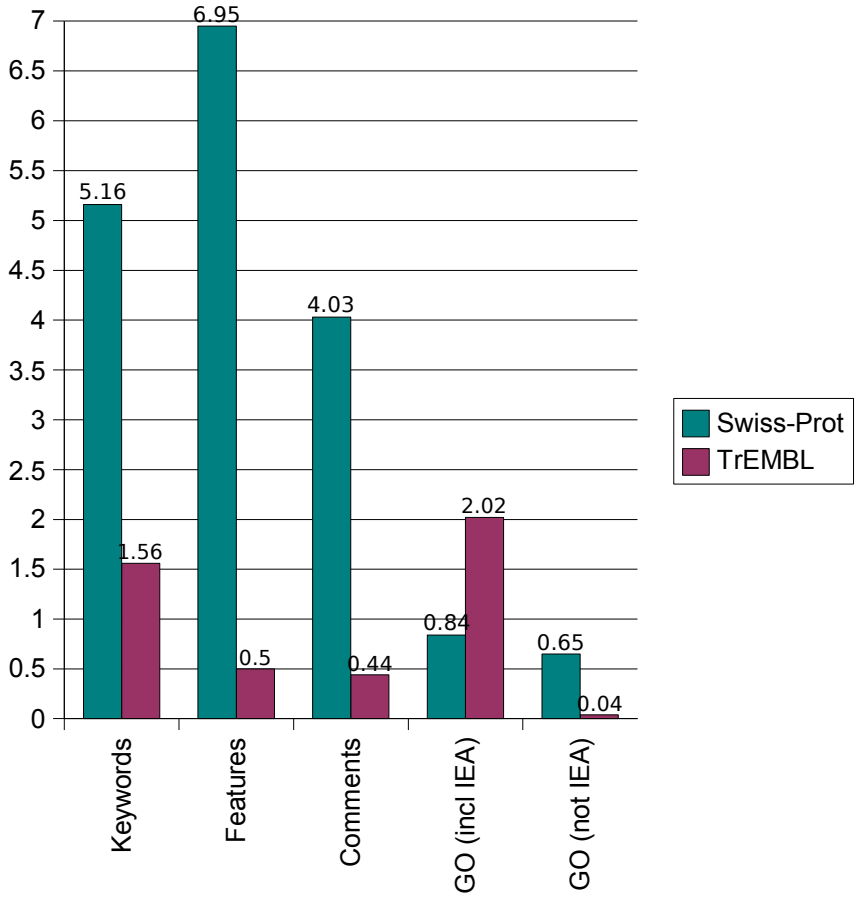

Figure 2.3: Average number of annotations per entry comparing Swiss-Prot and TrEMBL

This leads to the main task of the data mining team in the Sequence Database (SeqDB) group at the European Bioinformatics Institute (EBI), who are increasing the number of annotations in TrEMBL using common machine learning techniques. One outcome of their work is Spearmint, a Decision Tree (DT) algorithm that helps to extract knowledge found in Swiss-Prot. Rules generated for different types of annotations are checked for consistency by a system called Xanthippe, and all

remaining rules are then applied to TrEMBL [6]. Automatic annotation for Keywords, features and comments is implemented in this way [1, 7]. However the feature annotation pipeline is not yet in production mode. Figure 2.2 and Table 1 show that automatic annotation of Keywords is simple since they are in a controlled vocabulary. In contrast Swiss-Prot comments, which contain free text, are more difficult to predict, and this is reflected in a much lower protein coverage.

For most types of annotation in UniProtKB such as Keywords, features or comments, there is no source declaration attached directly to the database (or the flat file), so it is difficult to trace back how many of these annotations are derived solely by computation, and which are manually curated. The target for this work, GO terms, however does include an evidence code. The "**I**nferred from **E**lectronic **A**nnotation (IEA)" evidence code is applied whenever a GO annotation depends directly on computation or automated transfer of annotations from other databases. IEA is used when no curator has checked the annotation to verify its accuracy [8]. Going back to Figure 2.2 and Table 1, it is evident that 42% (105,301 entries) of all Swiss-Prot records have GO annotations, but only 13% (33,624 Swiss-Prot entries) have manually added or curated GO associations. Release statistics for Swiss-Prot state that there are approximately 0.82 GO terms per average record [4], 0.65 of which are manual GO annotations. In TrEMBL, 59% (2,070,472) of the entries have GO annotations, corresponding to 2.02 GO terms per entry [5]. The greater number in comparison to Swiss-Prot is not surprising because in TrEMBL all IEA techniques are included whereas Swiss-Prot only allows the most reliable ones. Only 1% (37,260 entries) of TrEMBL entries contain manual GO annotation giving an average count per entry of 0.04 non-IEA annotations (Figure 2.3). The evident gap between GO annotation and other annotation types is probably caused by the relatively late development of Gene Ontology in the year 1998, whereas Swiss-Prot was started in the early 1980s.

2.2 Prior Work

The relatively high portion of GO annotated entries in TrEMBL can be explained by the work of GOA, the GO Annotation Team at the EBI. GOA produces a huge amount of electronically inferred GO annotations for TrEMBL. They also do manual GO curation resulting in a number of GO terms which are NOT inferred from electronic annotation in addition to integrating manual annotations from other databases. Most of the automatically produced annotations are based on simple mapping

tables, for example mapping Swiss-Prot Keywords or InterPro[1] signatures to GO terms [10]. These mapping files are in part curated manually, which usually means that they are of reasonable quality although the number of false associations has only been counted on a very small sample set of data. As Camon et al. admit, InterPro group affiliation does not necessarily mean that all contained proteins have the same function [10]. There are some proteins that belong into a specific group but function differently from the other group members. Those "false positives" block an annotation of the specific GO term, and cause curators to turn to higher level terms. Hence those GO annotations inferred from an InterPro to GO mapping have to be treated with caution. Similarly, all more or less static mappings have to be treated with caution[2]. For this reason in this work only manually curated GO annotations will be taken into account.

Work in the field of automated protein or gene function annotation based on GO terms is not only conducted at the EBI. Elsewhere different approaches have been developed, mostly predicting GO terms for proteins based on sequence similarities. Uncharacterised sequences are searched against GO mapped databases and assigned GO terms of the best hits [11, 12]. A group at the DKFZ Heidelberg for example, applied Support Vector Machines (SVM) in combination with a homology search to transfer GO terms to unknown sequences and classify those predictions as true or false [13]. The system was trained on approximately 860,000 (organism wise grouped) sample sequences and applied to *Xenopus laevis* genes. The combination of multiple classification results (each organism set provided one classifier) produced predictions for about 50% of the Xenopus entries as well as a confidence value for each prediction. Predictive rule models were developed by Hvidsten and his group based on microarray hybridisation experiments [14]. The supervised learning methodology predicted 647 terms based on fibroblast response data for 517 genes. These attempts to predict GO terms, however, neglected the information contained in the hierarchical structure of GO and an approach taking the relations between GO terms into account was used by Barutcuoglu et al. [15].They started from protein interaction data, microarray expression data, collocalisation datasets and transcription factor binding sites to train multiple SVMs. The hierarchy of GO was introduced afterwards when a Bayesian Network was applied to verify the consistency of the predictions produced . This approach showed some improvements over mere machine learning. Frederick Roth's research group provided an approach using existing GO annotations and therefore not based on sequence similarity data and including the hierarchical structure of GO [16]. Using the FlyBase and Saccharomyces Genome Database (SGD) organism specific databases, they compared Decision

1 InterPro is a database of protein families, domains and functional sites in which identifiable features found in known proteins can be applied to unknown protein sequences [9].

2 mapping tables are checked and altered in response to user feedback but are not continuously adapted.

Trees and Bayesian Networks for the prediction of GO terms based on coexisting annotations of proteins. Attributes were selected such that they had no connection between them in the GO hierarchy so predictions were made only on completely different branches of the GO DAG. The result: Decision Trees outperformed the conditional probabilities of the Bayesian Networks at low false positive rates. A manual assessment of the resulting 100 predictions judged 41 of them as true and another 42 to be plausible.

These approaches were either only implemented for low throughput analyses and are therefore not applicable to UniProtKB, or were based only on sequence similarities, which can not take into account knowledge involved in manual curation. Although the additional knowledge contained in the complex relationships between GO terms is valuable and has been shown to improve predictions, the majority of the named approaches did not consider the hierarchy of Gene Ontology. Most approaches were also focused on a few organisms instead of providing general, organism-independent GO annotations. One problem with GO in UniProtKB, described later in more detail, is that there are no negative examples. None of the named approaches addressed this problem.

2.3 What is GO?

Free text descriptions of a gene or protein function tend to vary from person to person, from database to database and from organism to organism. Natural language provides hundreds of ways to describe the same fact, and this makes it particularly difficult to analyse automatically. Controlled vocabularies and ontologies such as the Gene Ontology (GO) is one way to deal with that problem.

An Ontology, in terms of Artificial Intelligence (AI), is meant to explicitly specify the objects contained in it. As Gruber defined in 1995, “an Ontology is an explicit specification of a conceptualization” [17]. The term originates in philosophy and means a systematic account of existence and the representation of what “exists”. This is a key feature in AI systems. The set of existing objects are reflected in the representational vocabulary with which a knowledge-based program represents the knowledge about the world [17].

The Gene Ontology describes gene products and consists of three different controlled vocabularies. In general, controlled vocabularies aim to provide a carefully selected set of terms, which are used to describe things in a uniform way. Ontologies, in extension to controlled vocabularies, also define the relationships among those terms and represent them in a hierarchy. The intention of GO is to address the need for consistent descriptions of the functions and subcellular location of gene products in different databases. Functional descriptions should be uniform and comparable so that

computational approaches can be applied more easily. The Gene Ontology is designed so that the terms provide species-independent descriptions. Hence GO terms are adequate for the annotation of molecular characteristics across organisms. The three domains of the Gene Ontology are Biological Process, Cellular Component and Molecular Function. Since GO is an ontology, GO terms are arranged in a complex hierarchy using two different types of relationships: “is-a” and “part-of”. GO also allows multiple children, as well as multiple parents, for a term [18]. The resulting directed acyclic graph (DAG) is exemplified in Figure 2.4. Red arrows denote “part-of” and blue arrows denote “is-a” relationships between the terms. The three disjoint domains of GO are also visible. Within the cellular component section of GO, the term “plasma membrane” is a child of “membrane” just like “membrane part”. But the relationships to their parents are different. “Plasma membrane” is a “membrane” whereas the “membrane part” is obviously a part of a membrane. Both child terms are, per definition, more specific than the parent term. The rather normal fact of having multiple children does not affect the uniqueness of paths from each term to the root node, but, as the graph shows, allowing multiple parents for a term does. The term “membrane part” for example has two parental terms, “membrane” and “cell part”. Hence there may be more than one path from a child term to the root.

The Gene Ontology website provides more details [8] and GO terms can be browsed with tools like AmiGO [19].

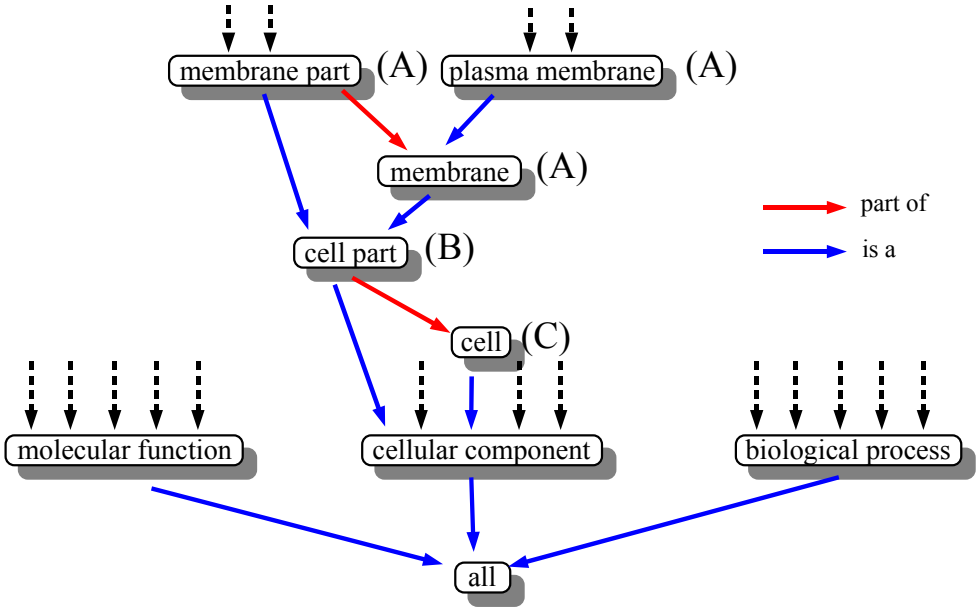

Figure 2.4: Visualization of an extract out of the GO hierarchy. Note that the most common term is at the bottom, the most specific term at the top

2.4 GO Annotations in UniProtKB

As mentioned before, only a small fraction of entries in UniProtKB have manual GO annotation. The GOA team at the EBI is continuously providing manual curation for Gene Ontology terms, but the number of entries in UniProtKB (and particularly in TrEMBL) is increasing at such a fast pace that manual annotation methods cannot keep up. Most experimental data is provided for model

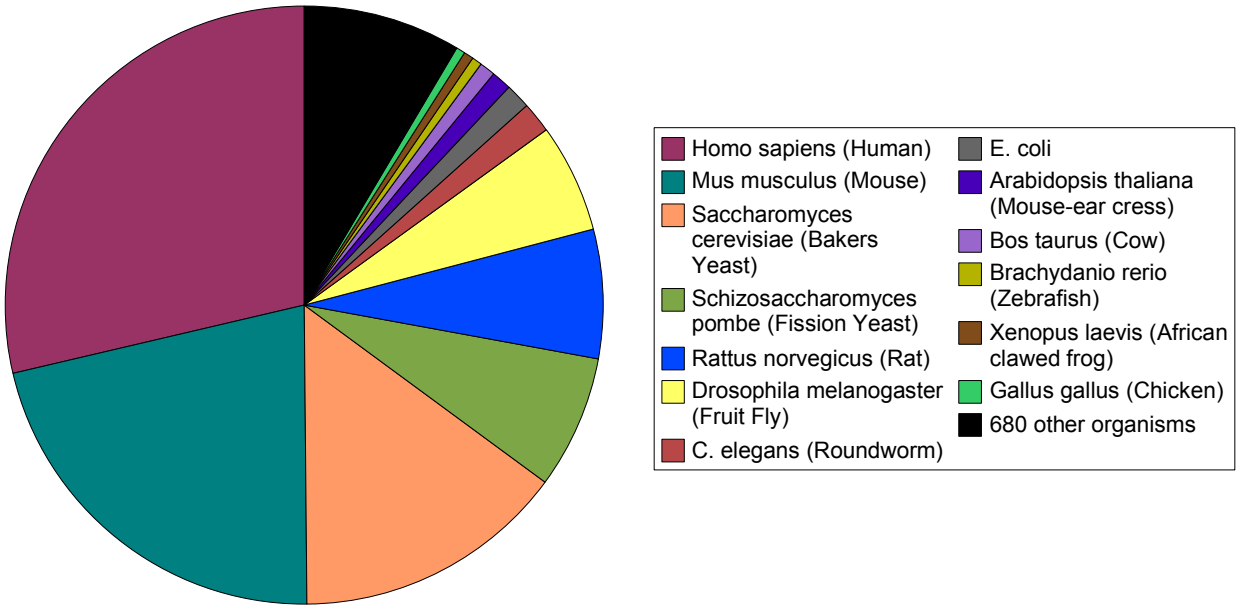

Figure 2.5: Distribution of organisms across database entries containing GO annotations in Swiss-Prot.

organisms due to both the used model organisms in the laboratories, and the major involvement of model organism databases. For this reason, GO annotation methods also focus their attention to those selected organisms. The graph of the distribution of GO annotations in Swiss-Prot (Figure 2.5) shows that 92% of all entries containing manual GO annotation belong to only 13 organisms. Whereas the remaining 8% of GO annotated entries are contributed by as many as 680 different species. This reflects the work of the model organism groups in the GO consortium. In addition to about 120,000 other organisms GOA itself is mainly responsible for human, cow and chicken, whereas mouse data is contributed by the Mouse Genome Institute (MGI). Bakers yeast is maintained by the Saccharomyces Genome Database (SGD), fission yeast originates from GeneDB, rat data from Rat Genome Database (RGD), FlyBase provides data for the fruit fly, WormBase for the roundworm, The Arabidopsis Information Resource (TAIR) is responsible for Arabidopsis data and finally the Zebrafish Initiative (ZFIN) provides zebrafish data. By far the most important

organisms in the graph shown are human and mouse, which account for more than half of all manual GO annotations in Swiss-Prot. This examination is based on absolute numbers of entries in Swiss-Prot. The universe in this case consists of only those entries containing at least one manual GO annotation.

Absolute numbers of records, however, do not reflect the number of records per organism, which correlates with the proteome size of each species. So the question is, how do GO annotations cover all recorded entries of an organism in UniProtKB? The percentages of Swiss-Prot entries containing manual GO annotations out of all records per organism are shown in Figure 2.6[1]. The organisms responsible for the major part of all GO annotated records in Swiss-Prot are marked in ruby. They are distributed over the whole range of coverages. For example the popular model organism *Arabidopsis thaliana* (Mouse-ear cress) has a coverage of only 7% even though it contains annotations from a model organism database. The maximum coverage of organism-specific proteins in Swiss-Prot is 76% for the baker's yeast (*Saccharomyces cerevisiae*). Organisms in turquoise contribute only a very small absolute number of sequences to all GO-annotated proteins in Swiss-Prot, but nevertheless their coverage with manual GO annotations based on Swiss-Prot entries varies between 8% and 37% (considering only fully sequenced organisms). Organisms whose genomes are not yet fully sequenced are depicted with dashed lines[2]. For these, the shown percentage of manually GO-annotated Swiss-Prot records does not reflect the coverage of the complete proteome, but only their coverage in existing database entries. Considering their complete proteome would result in distinguishably smaller numbers. For example, all the available entries in UniProtKB for *Tityus cambridgei* (an Amazonian scorpion), currently only 25 proteins, have manual GO annotations. These are components of its venom which were sequenced. Since about 60 components of this toxin were already separated by Mass Spectrometry methods [20], the whole organism is assumed to have a much greater count of proteins and the coverage of its proteome would be much smaller than 100%. The graph makes it clear that only a few of the model organisms, like bakers yeast, fruit fly, fission yeast, human and mouse are covered by more than 50% (up to 76%) with GO terms. But in other organism-specific sequence sets, even in some model organism sets, only about 5% have at least one GO annotation. One has to keep in mind, that this is only a Swiss-Prot related view, which is not reflecting genome/proteome wide GO annotation.

1 Note that organisms with less than 25 Swiss-Prot entries are not included in this graph.

2 The conclusion that they are not completely sequenced yet is drawn from their absence in Integr8. The team at the EBI provides a web site with easy access to integrated information about sequenced genomes and their corresponding proteomes. See http://www.ebi.ac.uk/integr8.

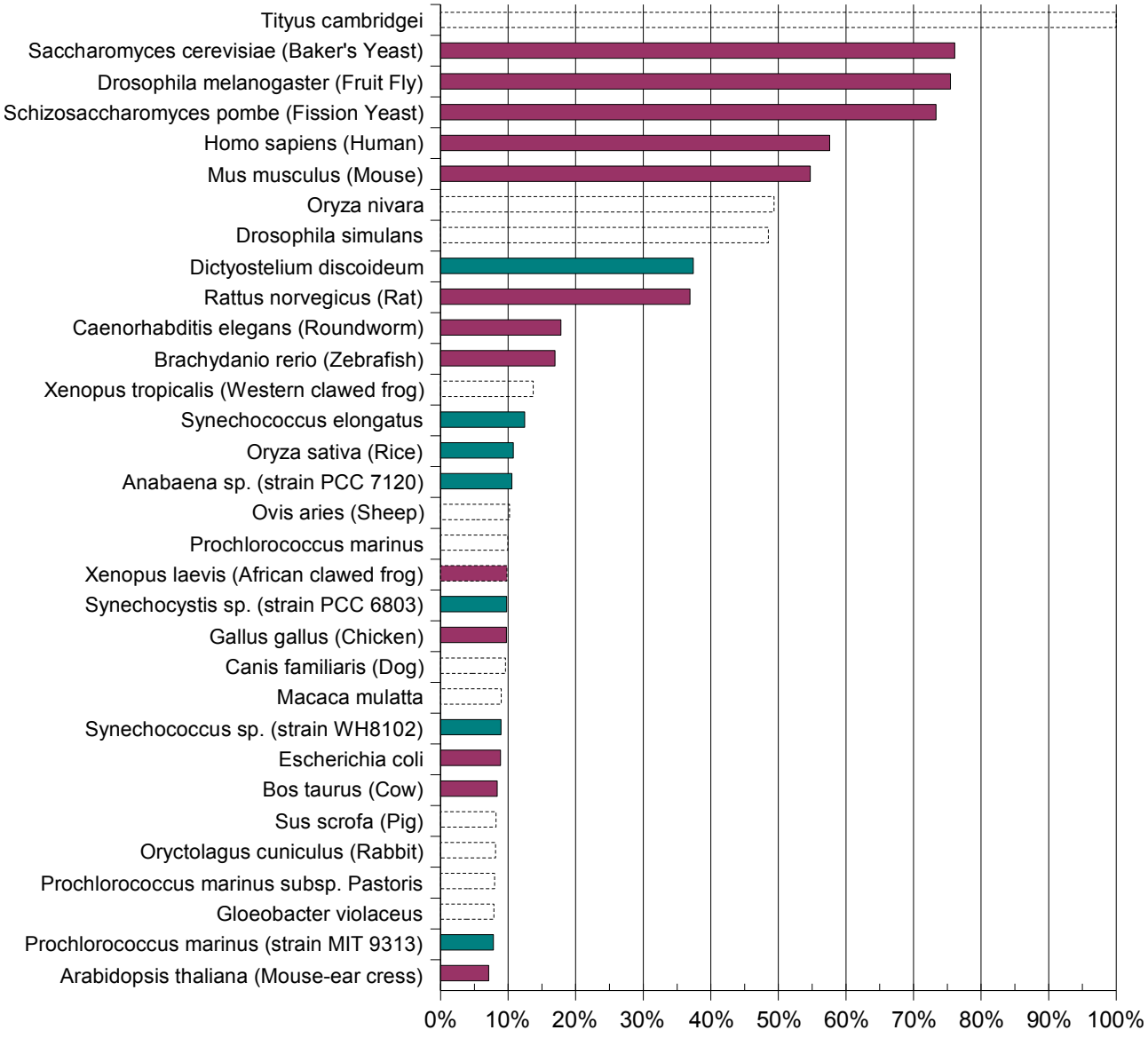

Figure 2.6: Percentages of GO annotated proteins in all entries per organism in Swiss-Prot. 13 organisms were marked ruby. Those are the organisms covering 92% of all GO annotated Swiss-Prot entries. Dashed columns are from organisms whose genomes are not yet fully sequenced.

To give an overview of how the situation is with respect to the whole of UniProtKB, Figure 2.7 shows the 13 most important organisms for GO annotation in Swiss-Prot. The basis for this examination are all UniProtKB records per organism. All but the African clawed frog (*Xenopus laevis*), which is only currently being GO annotated, have completely sequenced genomes and UniProtKB contains all known proteins of those organisms. The fraction of entries containing manual GO annotations out of Swiss-Prot and TrEMBL are shown as well as the percentages of entries NOT containing manual GO annotations in both parts of UniProtKB. The average coverage of those organisms in UniProtKB with GO annotations is 22% whereas considering only Swiss-Prot, the 13 organisms reach 35% as mean. Remarkable in this analysis is the fission yeast, which reaches a GO annotation coverage of 84% over all UniProtKB . Next are baker's yeast (69%) and the mouse (44%) followed by fruit fly (29%), rat (18%) and human (10%). Even underneath those lie *C. elegans* (8%), *E. coli* (5%), *Arabidopsis thaliana* (4%), chicken (4%), cow (3%), zebrafish (3%) and *Xenopus laevis* (2%). That shows that, even within such important model organisms, only the yeast is more than half manually GO annotated. This is because the SGD project established a

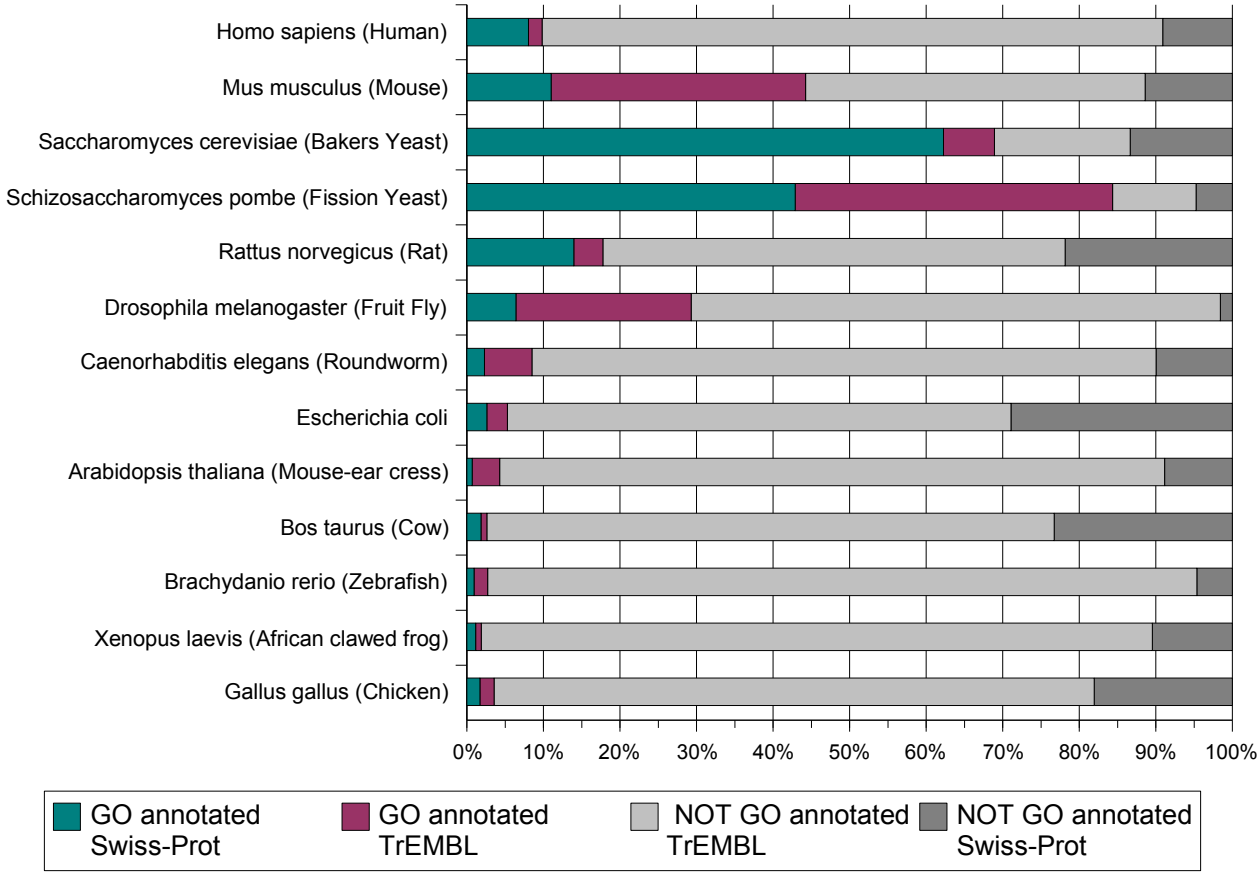

Figure 2.7: The 13 most important organisms for manual curated GO annotation in UniProtKB. Each bar represents all proteins of one organism (100%) in UniProtKB. Organisms are ordered by their absolute number of GO annotated database records in Swiss-Prot descending from the top.

well characterised database early on (1996) [21]. Noticeable in that figure is also that for some of those organisms great parts of manual GO annotations are not only found in Swiss-Prot but also in TrEMBL. For mouse, *Drosophila*, roundworm and *Arabidopsis,* approximately twice as many records (containing GO annotation) exist in TrEMBL in comparison to Swiss-Prot. This reflects that GOA and supporting database teams do not specifically focus on Swiss-Prot but annotate mostly independently. The philosophy behind the GO consortium curated annotation datasets is to preferentially annotate proteins without any GO annotations (which are found mostly in UniProtKB/TrEMBL) or as for GOA proteins which are specifically important in human health. In contrast, the distribution of records containing NO manually curated GO annotations is as Figure 2.2 shows: TrEMBL holds many more entries, lacking GO annotations, than Swiss-Prot does.

2.5 Aims of this Work

The main aim of this diploma thesis is to investigate the methods of automatically annotating Gene Ontology (GO) terms in UniProtKB/TrEMBL, to complement the efforts by the GOA team. The approach aims to provide a general, organism-independent, prediction mechanism that simulates manual annotation as reliably as possible. Machine learning algorithms on the basis of prior annotation algorithms, developed by the data mining group at the EBI, are therefore evaluated for their usefulness when applied on GO annotations. The knowledge about GO term associations is extracted from Swiss-Prot, taking advantage of the rich and high quality (manual) annotation there, with the intention of applying the resulting rules to TrEMBL. The basis of all predictions is not just sequence data or expression experiments, but uses a variety of (manual) annotations found in Swiss-Prot and TrEMBL. Predictions shall therefore be based on multiple sources of protein information rather than focussing on single attributes, trying to mine as much information contained in the database as possible. As a preparatory step, existing GO annotation will be enriched using the GO hierarchy without changing the original meaning of the annotation. An approach to generate negative GO associations is used to achieve a more complete knowledge base for the machine learning algorithms. To ensure the best possible results, statistical analysis of the UniProtKB database concerning non-existing GO terms is included. This way, the problem of missing statements about negative GO terms will also be addressed. Since prior work showed the GO hierarchy to be helpful for improving predictions, this work also evaluates the potential for taking

the GO hierarchy into account during rule extraction. In comparison to the mappings of the GOA group, the described approach takes false positive records into account and attempts to simulate manual annotation of protein data. The algorithm will provide information about the accuracy and confidence of extracted rules.

2.6 Difficulties

The introduction of machine learning algorithms raises the need for a good set of data as a learning base. Data from which rules are to be learned have to be consistent and preferably complete. Take a look for example at a given set of five entries describing people. If three of them are male, the conclusion for the other two is obviously that they are female. This conclusion however, is derived from the assumption that all of the five people were taken into account while classifying males. If the person who was responsible for that classification simply forgot to include the last two people, the conclusion that they are female no longer holds true and nobody knows if they are male or female. Now what happens if one nevertheless tries to derive rules about those five people, assuming that the classification was complete? A rule like “all males have short hair” could be derived. But if the persons classified as females are actually also males and have long hair, these rules would be wrong. The quality of those rules cannot be assessed as long as the knowledge on which they are built is incomplete.

A similar problem occurs with GO terms in Swiss-Prot. Only 13% of all Swiss-Prot records contain at least one manually curated GO annotation, and it is very improbable that all the other proteins in that database do not have any function which could be represented in GO terms. It makes more sense to assume that nobody tried to add GO terms there. But knowing that makes it clear that NO conclusions can be derived about proteins where NO specific GO term has been added. To address this problem, all proteins without any GO terms were excluded from the learning set. Yet even if only those proteins with at least one GO annotation are selected as a training set, there is another problem. There are over 20,000 distinct GO terms, describing Biological Function, Cellular Component and Molecular Process of a protein in great detail. It is a non trivial task to consistently annotate within the Gene Ontology. Also there are a lot of different curators annotating GO and each of them has his/her own knowledge and preferences concerning terms. Considering only one specific GO term out of the Biological Process domain for the moment, there are different GO terms describing a Molecular Function or a Cellular Component which could be used in combination. Also very often, GO terms of different granularity are applicable to a protein, so it

depends on the curator and of course the experiments or analyses carried out, as to which of those GO terms will be added and which will be omitted. The result is very inconsistent GO annotation in UniProtKB. Once again (also in the set of proteins containing only records with at least one manual GO annotation), not a single conclusion about a GO term NOT assigned to a protein can be made, even if only proteins with at least one GO annotation are examined. For this reason an algorithm to determine true negative GO annotations is vital to any GO prediction algorithm applied to Swiss-Prot.

Addressing the problem of missing negative examples, however, does not yet solve the issue of inconsistent and incomplete positive GO annotations in Swiss-Prot. As mentioned before, the available knowledge about proteins differs and the policy of GO annotation states that only terms the annotating person is absolutely confident about will be included. Additionally the curated data of a specific protein will generally not be updated (due to the overwhelming amount of still not annotated data) so that experiments after the date of annotation, will not be included. These facts and the huge number of detailed GO terms, many of which are very similar, cause GO annotations to vary in accuracy, even if the actual function of two proteins is the same. Unlike taxonomy annotation, GO annotation is not complete. Taxonomic classification of a protein in UniProtKB always includes the entire path from the most detailed term (a specific organism in that case) to the root (one of the five domains of life: Archaea, Bacteria, Eukaryota, Viruses or Viroids). With GO terms, however, only the most detailed term is assigned to a protein and the path through the hierarchy of Gene Ontology is not included in the annotation although curators do keep in mind this information during the annotation process.

3 Methods

3.1 Spearmint on GO terms

Despite the fact that GO terms are arranged in a complex hierarchy, there is some overlap with Keywords in Swiss-Prot. The terms, contained in a controlled vocabulary in both cases, describe the Molecular Function(s), Biological Process(es) or Cellular Component(s) in which a protein normally acts in a human readable form. It therefore stands to reason that an algorithm which successfully predicts Keywords is also, in an adapted form, usable for GO terms. As mentioned before, the data mining group at the EBI has developed a machine learning process, automatically predicting Swiss-Prot Keywords on TrEMBL, based on knowledge filtered out of Swiss-Prot. The system used is a Decision Tree implementation embedded in a State Machine.

C4.5 Decision Tree

The Decision Tree algorithm is a popular machine learning concept. It aims to describe an existing set of objects in terms of a selected target attribute. The main idea behind it is the “Divide and Conquer” mechanism. The original set of objects is divided by a decision based on one attribute of the objects. Each subset is then again divided until every subset is uniform and holds only objects of one type. The quality of a Decision Tree is measured by its length, i.e. the number of levels it has. Therefore a good Decision Tree tries to classify all objects of the universe with as few decisions as possible. The example in Figure 3.1 shows this for a set of five people. Three of them are male (Jack, Bert and Bob), and two are female (Anna and Maria). The task of the Decision Tree is to figure out what distinguishes the male from the female. It is provided with three attributes describing the people, one attribute is describing the length of their hair, one their finger nails and the other one tells if a person has glasses or not. A very simple algorithm randomly picks one of those attributes, lets say “long hair”, and divides the group of people into two subgroups, those persons having long hair (this branch is marked with a '+' in the graph) and those NOT having long hair (branch marked with a '-'). Since the first subgroup (containing Anna, Jack and Maria) is still not uniform, the division has to be repeated. Again picking a randomly chosen attribute, let's say “painted nails”, results in two more groups. This time every subset is uniformly male or female and the algorithm terminates. But this is not necessarily the Decision Tree describing the dataset in the shortest possible way. In our data for example was one attribute, the “glasses”, describing male and female persons on its own. The selected dataset would be divided into uniform groups by only one decision using this attribute. But finding the ideal Decision Tree for a set of data is a non trivial

	Long hair	Glasses	Painted nails	Female
Anna	Yes	No	Yes	Yes
Bert	No	Yes	No	No
Bob	No	Yes	Yes	No
Jack	Yes	Yes	No	No
Maria	Yes	No	Yes	Yes

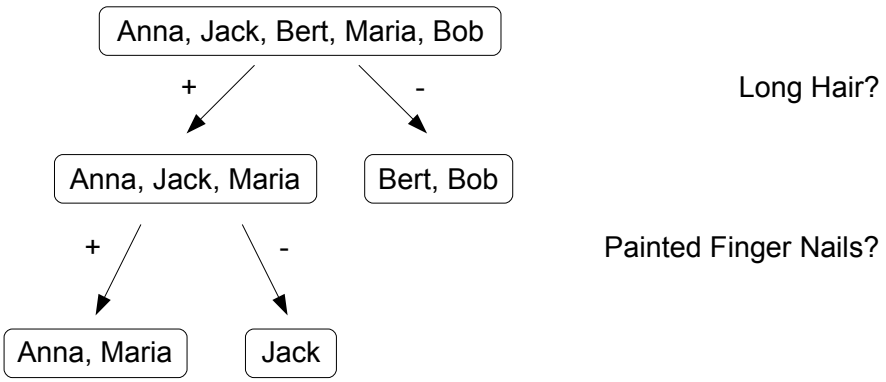

Figure 3.1: A simple example for a dataset and one possible (not ideal) resulting Decision Tree. The rule extracted here is: IF long hair AND painted nails THEN female

problem, especially considering big datasets with many attributes. The number of possible Decision Trees for one dataset is finite but very large so that trying all those possibilities would take too long for big data sets [22]. Several heuristic approaches try to solve this problem by producing a Decision Tree that is not necessarily the best one for a given dataset but in general performs reasonably well on real datasets. One of them is the very popular Decision Tree induction algorithm called ID3 [22]. It maximises the "information gain" for each decision. The attribute achieving the greatest information gain will be chosen for the next division of the universe. Information gain takes into account the relative numbers of each target type per resulting subgroup. Since all male persons in our small example wear glasses and the female do not, the test on attribute "glasses" has a much higher information gain than the test on "long hair" which results in a group containing both types of persons.

However, ID3 is not suitable for all datasets. One major drawback is that it can only handle binary attributes. But it is easy to find examples of non binary, numerical attributes. Describing people is for example much easier when including continuous attributes like size or weight. A quite simple enhancement of ID3 to address this problem is the C4.5 algorithm [23]. This sorts the objects in the

universe according to their numerical value for the current attribute and finds the best threshold to maximise the information gain when dividing the universe into a group with lower or equal values of the attribute and a group having higher values of this attribute.

Another problem all Decision Trees have in common is that of over fitting. It is possible that the described algorithm produces Decision Trees which describe each single object of a set in its own group of size one. A Decision Tree would then have many long branches instead of a rather bushy outline, so the achieved knowledge representation is not as comprehensive as desired. A good representation of the information contained in a dataset groups similar items of the universe into as big as possible, yet distinctive subsets. The basic C4.5 algorithm can be extended to solve this problem in a very simple way. For reasonable Decision Trees, a minimum number of instances per node is introduced. Splits not fulfilling this condition will not be conducted.

An additional measurement of the quality of a tree is its classification ability. Therefore, the number of true and false classifications at all nodes using the training set are evaluated. But considering only instances the learning process the tree was based on, does not necessarily represent the behaviour of that tree with unknown data. The general approach to that problem is simply withholding some of the known instances and training the Decision Tree on the remaining ones. The excluded part of the known instances is subsequently used for testing and measuring the quality of the resulting classifier. One problem with this approach is that the learning process will then be based on a lower number of instances and not on the entire knowledge available. In small datasets this can lead to too small training sets not representing the universe any more. A common method to address this issue is Cross Validation. This is based on repeating the above “holdout” method dividing the dataset into a number of equal groups. Training and testing then takes place iteratively, complementing the knowledge from each training set by changing the test group with each iteration. So in the end, every instance is used for learning and also for testing.

To measure the quality of a single node, a confidence value is calculated using the numbers of correctly and incorrectly classified training instances. Table 2 shows the confusion matrix, the definition of True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN)

	Classified as true	***Classified as false***
True	True Positive	False Negative
False	False Positive	True Negative

Table 2: Classification of correct and incorrect predictions

classifications. A classification is True Positive if an instance was classified as true and also the real value of the examined attribute is true. The opposite applies to True Negatives. If an instance was classified as true but is really false, it is called False Positive and if the classification was false but the real value was true it is a False Negative. The aim of any machine classification algorithm is to achieve many true classifications while maintaining a low number of false classifications.

These named values represent the classification of the known dataset. What really interests, however, is how the Decision Tree performs on unknown data. But naturally it is not known whether a classification of an unknown instance is correct or not. So the performance on new data is estimated, based on the performance with the known test set. The derivation of this confidence value is described in detail by Witten and Frank [24]. The confidence of a Decision Tree has to be based on the number of instances in the test set. The more instances that were tested on the tree, the greater is the probability that the observed performance also describes the performance on unknown data quite well. Formula (1) shows the calculation of the confidence value. It is derived from *p*, the precision of the node (Formula (2)), *n* is the number of instances at the current node classified as positive (3), and a constant *z*. Note that negative classifications were ignored here because the primary aim is to predict, not to contradict, GO terms even if false negative classifications represent missed true positives.

$$confidence = \frac{p + z^2 - z * \sqrt{\frac{p}{n} - \frac{p^2}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}} \quad (1)$$

$$precision = p = \frac{TP}{FP + TP} \quad (2)$$

$$n = TP + FP \quad (3)$$

The constant *z* is dependent on a threshold which expresses the percentage of cases where the real performance is correctly represented in the training environment. So the confidence value using z for 95% stands for the relation of TP to FP on a node which will be observed in 95% of all cases where the tree is applied in the real world. A confidence value of 1 therefore means that in 95% of all cases only True Positive classifications are assumed to result from the current tree with real (unknown) objects. A value of 0 is the worst case, in which in the real world only False Positive classifications will be expected in 95% of all cases. By having this confidence value for each node,

it is possible to introduce quality controls. So the existing algorithm (implemented in the Kraken framework which was used for this work) excludes all nodes with a confidence less than a specified threshold.

Even with all those extensions, Decision Trees are computationally relatively inexpensive, considering the training phase as well as in terms of application. They are capable of dealing with large datasets, which is one reason why they are suitable for high-throughput Keyword prediction on Swiss-Prot. Rules extracted by the algorithm are easy to understand and human readable, which makes them easy to revise and evaluate for curators. But there are some drawbacks as well. Decision Trees are not able to handle non-categorical target attributes and too many, especially numerical, attributes can lead to very complicated trees. Another disadvantage of Decision Trees is that they can only separate linearly, which leads to a lot of incorrect classifications on non-linearly separable datasets. They also work best with datasets where negative and positive examples are roughly equally distributed.

Spearmint Pipeline

Spearmint for Keyword prediction uses the described Decision Tree, working with several protein attributes found in TrEMBL. Since the rules generated shall be applied to TrEMBL, only attributes also found there can be used. General information found in every TrEMBL entry includes taxonomic classification, InterPro group and InterPro matches. InterPro is a resource for protein families, domains and sites and currently combines 10 protein signature databases including Pfam, ProDom, PROSITE and TIGRFAMs. Signatures describing the same protein family or domain are grouped into unique InterPro entries with an accession, description and some cross-references as well as annotations [9, 25]. An InterPro group therefore refers to one of those protein families or domains whereas the InterPro match means a single match of a signature. Those attributes are used as attributes for the Decision Tree targeting Keywords.

This machine learning algorithm is embedded in a pipeline of processes in the form of a state machine. Several states, each processing a product in a specific way, can be put together in varying order. This way, loading protein data, preprocessing them, splitting, learning and testing can all be done in a separate state, each delivering its result to the product and forwarding it to the next state. This results in a very flexible approach to the data mining process. If for example the dataset has to be changed, only the loading state has to be replaced. Or for using a different splitting algorithm instead of Cross Validation, only the splitting state is affected. A typical Keyword prediction state machine is schematically shown in Figure 3.2. A loader state is in charge of loading the known dataset into a product, which will be processed by the following states. To achieve a high quality

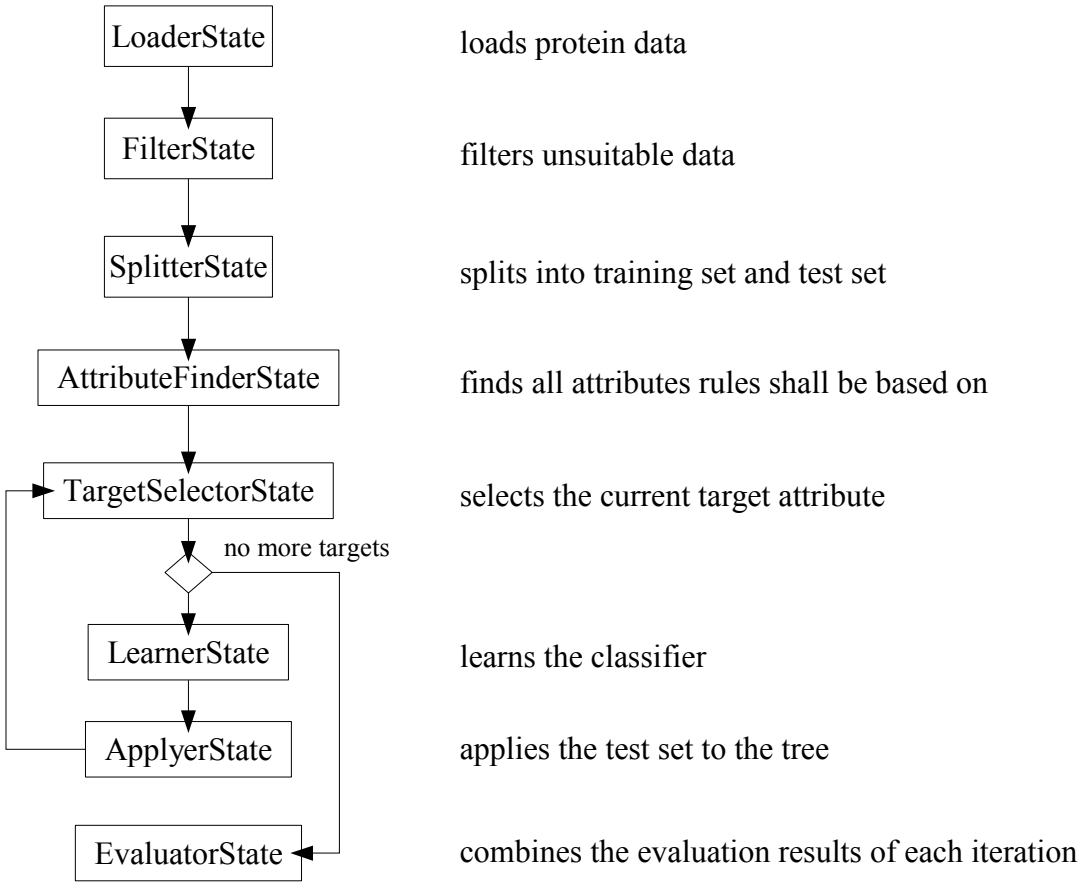

Figure 3.2: A simple sequence of states for data mining. This work flow does not include Cross-Validation.

learning base, a filter state filters all fragmented or hypothetical proteins. Then the splitter state divides the dataset into a training set and a test set of protein data. The Attribute finder state assembles a list of available attributes out of the dataset. This process can be narrowed to some desired attribute types (for example TrEMBL attributes). Since a Decision Tree has to be generated for each target (i.e. for each single Keyword in our case), a target selector state extracts all existing target values and selects one per iteration. For this target, a learner state (the C4.5 algorithm in our case) builds a classifier which is supplied with a confidence value and then tested in the applier state with the hold back test set. During this process, predictions and contradictions are added and stored with the proteins where applicable. The next step is to go back to the target finder state and repeat the learning and application step for each remaining target. When all target values have been learned and the resulting trees have been tested, the evaluation state summarizes the performance of the algorithm over all classifiers with the help of the product. This is now holding all information about predictions and contradictions as well as the original information of each protein. Note that in this simple example no Cross Validation is included.

For simplifying the adaptation of Spearmint, every action on a ProteinData object is modelled as a Command. These Commands have an execute method which takes an Object of some type and performs specified actions on it. The Java 5 option of using generics[1] is very helpful at this point. It enables the Commands for example to also work on Collections of various Java types. But additionally in Spearmint all those execute methods have a specified return type. Commands for testing, e.g. whether a protein has a specific GO term annotated, return booleans. The procedure of using Commands is similar to a known design pattern in programming, the "Visitor Pattern" [26].

GO Integration

In order to adapt the existing system, GO terms had to be integrated and retrieved from the database first. The existing Java class holding all information of one UniProtKB entry (ProteinData) only contained part of the GO database cross-references. Hence a new system to integrate GO annotations with all their information was implemented. EBI internally maintains a database table of protein to GO relations, which is updated regularly from the Gene Ontology database. Amongst other columns, this table also contains one column for UniProt accession numbers, one for corresponding GO ids and one for the evidence code of each GO association. This table is read in automatically and each line is transformed into an instance of the Java class GoAnnotation. Those GoAnnotations are grouped into lists per UniProt accession number and stored in a Java HashMap having the accession number as key and the list of GoAnnotations as value. This HashMap is then stored as a serialised GO annotation cache for easy access through the ProteinData. The whole procedure is depicted schematically in Figure 3.3. With the help of the resulting cache it is very easy to retrieve GO annotation information for each protein (by loading the cache into memory and querying it with the UniProt accession) and to store those GoAnnotations in the existing ProteinData object when needed.

Once the GO annotations were included in the ProteinData object, it was possible to adapt the Spearmint algorithm for GO prediction. An additional filter Command was implemented to filter out all proteins not containing any GO annotations. For the remaining proteins it was assumed that a GO term not annotated is a negative statement about the function of a protein. So the mentioned problem of missing negative associations was ignored in this first approach. Also the TargetSelectorState (remember Figure 3.2) was adapted to find all distinct GO terms in the dataset. The modular construction using states and commands made this simple. With the aim of comparing Keyword and GO term prediction with Spearmint, a fairly small training set of proteins was chosen.

1 Generics allow a type or method to operate on objects of various types while providing compile-time type safety. It adds compile-time type safety to the Collections Framework and eliminates the drudgery of casting. See http://java.sun.com/j2se/1.5.0/docs/guide/language/generics.html for more information.

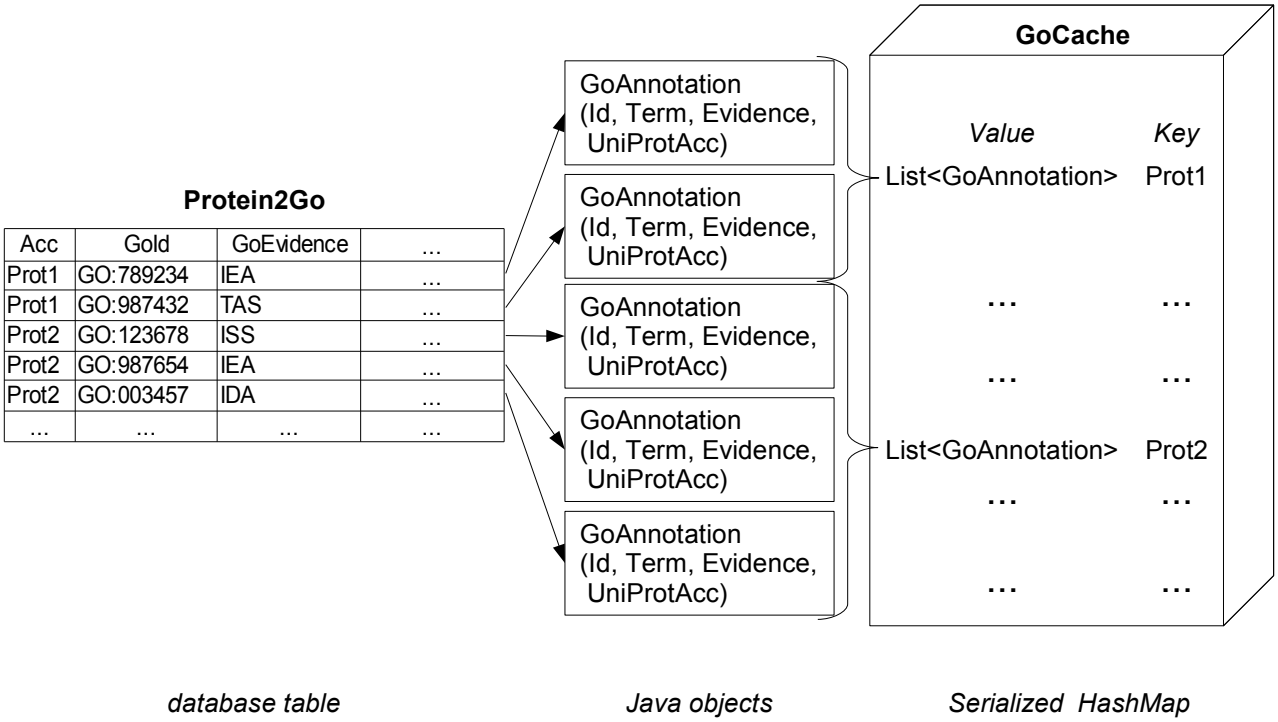

Figure 3.3: The simplified build procedure for the GO annotation cache.

Two large InterPro groups, Kringle proteins (IPR000001) and Connexins (IPR000500) containing 78 and 85 Swiss-Prot proteins were used. After the filtering step (filtering proteins without GO terms, hypothetical proteins and fragments) 63 proteins remained. With those, a standard Spearmint run was started for Keywords as well as for GO terms.

3.2 Simple Mapping

The Spearmint Decision Tree algorithm could, in comparison with the Keyword run, not predict GO terms on the selected training set of proteins with a sufficient quality (detailed results described in Section 4.2). Therefore an alternative approach had to be found. Swiss-Prot Keywords and GO terms are very similar even if the Gene Ontology has a far wider variety of terms. On some Swiss-Prot entries, the annotated Keywords are identical to the GO terms. Proteins for example can have the Keyword 'Gap junction' and also the GO term 'gap junction' (GO:0005921) attached. Hence a first idea was to use the obvious relationship between them and to investigate how a statistical approach performs in comparison with the GOA Keyword2GO mapping.

The aim was to find out which manually annotated GO terms in Swiss-Prot appeared repeatedly in conjunction with a specific Keyword on a protein. To detect the direct relationships between Keywords and GO terms and to convert them into reliable rules, Bayesian statistics were applied. The Bayesian rule for conditional probabilities, shown in formula (4) allows the calculation of the probability of a GO term conditioned on the existence of a Keyword annotation. This formula represents the probability that a specific GO term, let's say 'gap junction', will be annotated whenever a specific Keyword, say 'Gap junction', is already annotated. The pipe '|' in the formula denotes 'on condition that' and the notation *P(X)* is read as the probability *P* of an event *X*.

$$P(GO|keyword)=\frac{P(keyword|GO)*P(GO)}{P(keyword)} \quad (4)$$

As seen in (4), the Bayesian conditional probability can be computed with the help of three different probabilities. These are the probability of a Keyword being present if a GO term is already annotated, P(Keyword|GO), the probability of the specific GO term annotation being present in all proteins, P(GO), and the probability of the specific Keyword in all proteins, P(Keyword). This formula can be rearranged to one which only needs two of those probabilities, using the fact that the probability of having a GO term and the probability of NOT having the GO term annotated sum to 1. Formulas (5), (6) and (7) show the conversion of this fact into a form, which is then used to divide formula (4). Negation of a variable is shown through a horizontal line above the variable name.

$$P(GO|keyword)+P(\overline{GO}|keyword)=1 \quad (5)$$

$$\frac{P(keyword|GO)*P(GO)}{P(keyword)}+\frac{P(keyword|\overline{GO})*P(\overline{GO})}{P(keyword)}=1 \quad (6)$$

$$\frac{P(keyword|GO)*P(GO)+P(keyword|\overline{GO})*P(\overline{GO})}{P(keyword)}=1 \quad (7)$$

Since division by 1 does not change the original value, the result of that division still represents the conditional probability of a GO term. This is shown in formula (8). Now it is obvious that P(Keyword) can be cancelled from that fraction. What remains is the calculation of the conditional probability without the need to calculate P(Keyword) and only using P(Keyword|GO) and P(GO),

whilst taking the corresponding negative probabilities into account (9). The probability of a specific Keyword, which is not easy to calculate, is no longer necessary and all other probabilities can be estimated by counting directly from the dataset.

$$P(GO|keyword)=\frac{\frac{P(keyword|GO)*P(GO)}{P(keyword)}}{\frac{P(keyword|GO)*P(GO)+P(keyword|\overline{GO})*P(\overline{GO})}{P(keyword)}} \quad (8)$$

$$P(GO|keyword)=\frac{P(keyword|GO)*P(GO)}{P(keyword|GO)*P(GO)+P(keyword|\overline{GO})*P(\overline{GO})} \quad (9)$$

The same result may also be achieved by simply counting all proteins where the Keyword is annotated, and then out of these counting the proteins where the GO term is also available and divide the latter by the former. Table 3 shows this for a very simple example. For each (mock) protein the existence of a GO term X and/or Keyword Y is marked by a tick. Proteins one to five have the GO term annotated, and proteins two to five also have a Keyword association. Only a Keyword annotation is found for proteins nine and ten. By counting the co-occurrence of GO term and Keyword and dividing this by the number of Keyword occurrences, we get a result of rounded 0.667 for the probability of having the GO term X annotated, if Keyword Y is already present. Using the derived formula to calculate this probability yields the same result (10).

Protein	GO X	Keyword Y
1	✓	-
2	✓	✓
3	✓	✓
4	✓	✓
5	✓	✓
6	-	-
7	-	-
8	-	-
9	-	✓
10	-	✓

Table 3: An example distribution of GO term X and Keyword Y on ten proteins

$$P(GO\,X|keyword\,Y)=\frac{\frac{4}{6}*\frac{5}{10}}{\frac{4}{6}*\frac{5}{10}+\frac{2}{6}*\frac{5}{10}}=\frac{4}{6}\approx 0.667 \quad (10)$$

The advantage of the derived formula is that it is extendible to more than one condition. It is possible to calculate the conditional probability of a GO term depending on more than one annotation in the future. Therefore a conditional probability for each additional attribute, like P(comment|GO) or P(comment| $\overline{GO}$) respectively for comments, for example, had to be inserted into the formula (9) at each product. This would then look like formula (11). One has to keep in mind, however, that this calculation assumes independence between all used attributes, which does not represent the true state of the data. But such assumptions perform, in general, reasonably well on real data. The formula however now contains three more calculation steps and three more counting steps for just one additional attribute. It is easy to see that this leads to a time consuming and computationally expensive calculation, if a greater number of attributes is to be taken into account.

$$P(GO|A+B)=\frac{P(A|GO)*P(B|GO)*P(GO)}{P(A|GO)*P(B|GO)*P(GO)+P(A|\overline{GO})*P(B|\overline{GO})*P(\overline{GO})} \quad (11)$$

The calculation (adapted from Witten and Frank, 2000 [24]) also shows, that not only positive examples but also negative examples (where no GO term is annotated) are taken into account. What also emerges from the small example is that cases where the GO term is annotated but the Keyword is not, do not influence the result. Keeping the much greater variance of GO in mind, however, one can assume that a single Keyword is often expressed through more than one GO terms. Hence, the examples where a Keyword is not associated with the current GO term are not as important as the converse, because it may possibly be covered by another GO term. Thereby the disadvantage of this method is that lots of information contained in the database is likely to be discarded. Simplistic approaches like this one are, however, still often quite successful.

Using this calculation for each possible combination of a Swiss-Prot Keyword and a GO term leads to rules of the form: IF Keyword Y THEN GO term X, and provides a measurement for the reliability of each possible rule. A set of Keywords, leading to a minimum probability of 80% is then selected for each GO term, and a protein associated with one or more of those Keywords gets the GO term predicted. Contradictions of GO terms are made for every protein which have none of the Keywords attached. In comparison to the GOA Keyword to GO mapping, this approach is not static. This means that information found in “false positive” examples, where a GO term is not annotated but the Keyword is, are taken into account during the calculation and decrease the confidence value of a rule.

The naïve Bayesian principle works for different kinds of annotations. Since Keywords in TrEMBL are mostly derived by automatic annotation, and are therefore not manually curated, their usage could easily lead to a proliferation of erroneous annotations. InterPro groups were therefore also tested to predict GO terms in the same manner. An advantage of the described principle is that each rule is provided with a confidence value which makes it easy to pick only reliable rules.

3.3 A Complementary Approach

Since the Decision Tree approach left room for improvement and Simple Mapping tends to be rather complex for more than a few attributes, a complementary method was considered. Machine learning algorithms rely on the quality of the dataset on which they are based. Chapter One of this thesis already describes the drawbacks of the current state of the database, namely that GO terms are inconsistently annotated in UniProtKB and still have poor coverage. The hypothesis of the following approach was that a more consistent training set can enhance GO term predictions.

Positive Enrichment

The level of detail of GO terms assigned to similar proteins tends to vary. However, they often originate in the same lineage or are related to each other by only a few steps in the GO graph. Only the most detailed adequate GO term per protein is annotated in UniProtKB, so the relationships between the entries are not immediately obvious. Consider for example a set of proteins that have GO annotations of one and the same lineage. Any machine learning algorithm using only the existing annotation will have difficulties to find the information hidden in there. The group of proteins will not be detected as having comparable function and no rule will be built for it. But it is desirable to at least find a rule grouping those proteins under the most specific common term. Therefore a complete annotation, in terms of annotating the whole path from the detailed terms to the more general ones is needed. This raises an additional problem: considering the two different relationships in GO, paths from a specific term to a more general one do not necessarily mean that the specific term is an instance of the more general term. It is also possible that a path is interrupted by a “part-of” connection which results in a term not being an instance of the higher level term any more. For example, taking a look back at Figure 2.4 it is not correct to assume that “membrane” is a “cell” but “membrane” is part of a “cell”. Considering all (consecutive) child terms (A) as instances of the most general term (B) in a path of only “is-a” connections, and also imagining this parent term (B) to be a “part-of” child of some other term (C), it is always true that all children (A) of the term (B) are “part-of” term (C).

In terms of protein annotation however, the situation becomes more difficult. Imagine for example that the term “plasma membrane” is annotated to protein X. Using the rule above would allow you to annotate “part-of” “cell” to that protein as well. But since the GO annotation in UniProtKB does not include these qualifiers, the information about the connection between GO terms will be dropped. On the other hand, annotating the GO term “cell” without the qualifier to a protein which is originally only found in the plasma membrane of a cell is wrong. Taking a look at “is-a” relationships in comparison shows that it is possible to additionally annotate a parent term without changing the original meaning of the annotation. The protein above for example is found in the “plasma membrane”, which is-a “membrane”. So it is also true to say, that the protein localises to a membrane. Adding terms of higher levels is therefore allowed for “is-a” parents but not for “part-of” parents if the meaning of an annotation is to remain unchanged. Is has to be mentioned that since January 2007, Gene Ontology is “is-a complete”. That means that every GO term has now at least one “is-a” parent and therefore a complete “is-a” path to the root.

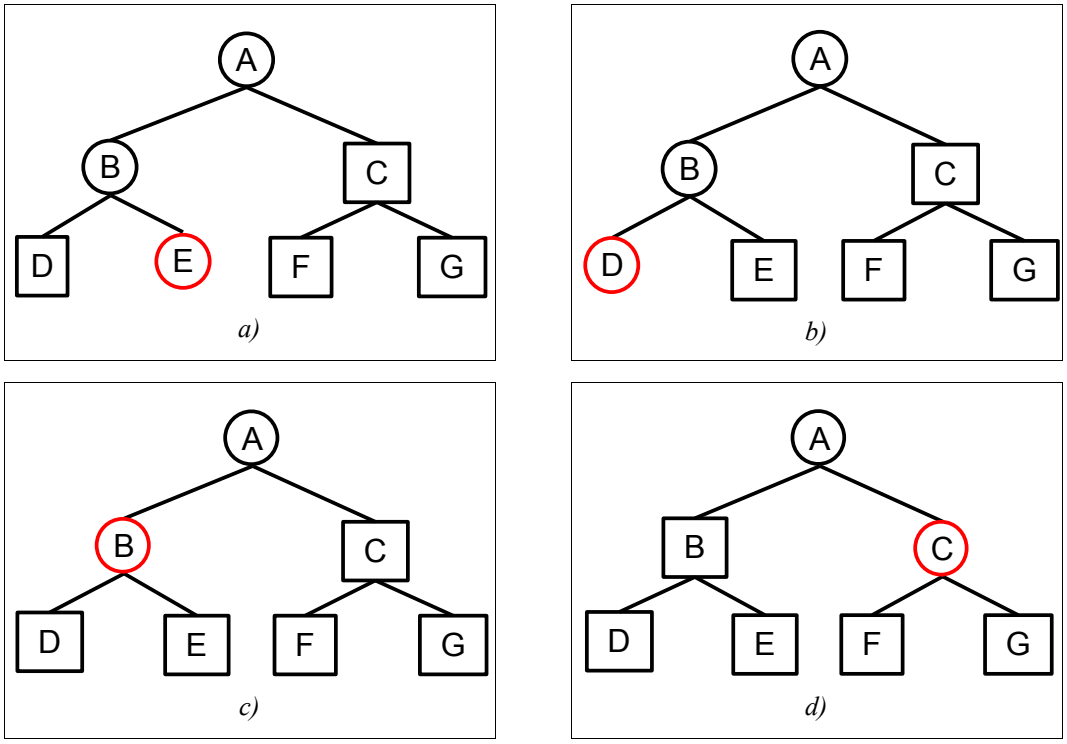

Figure 3.4: Four simple examples for proteins functioning similarly. Each tree simplifies the GO hierarchy. The originally annotated term is marked by a red circle. Black circles denote parental terms which can be additionally annotated without changing the original meaning of the annotation. Note that only “IS-A” relationships between GO terms are considered at the moment and the multiple parent characteristic of GO is not represented here.

Hence the positive enrichment approach aimed to complement the existing annotation using the GO hierarchy. Figure 3.4 shows what this means (note that only “is-a” connections are considered here). Four similar proteins have different GO annotations (denoted by a red circle) but those annotations are not far apart from each other concerning the hierarchy of the GO DAG. Proteins a) and b) are annotated at the same level of detail with sibling terms, but parental terms of their original GO annotations (black circles in the graph) are the same. For protein c), a more general annotation was made, since knowledge about that protein is not as detailed as for a) and b). However considering the parental terms in cases a), b), and c) shows that all three play a role in a process B (when considering only biological process, the same is true for cellular component or molecular function), and also an even more general process A. Protein d) has a GO term in a different branch annotated, but it still has process A in common with all other proteins when parent terms are considered. A machine learning algorithm should be capable of extracting the information that proteins a), b), and c) play a role in process B. Even though the rule is not as detailed as if it predicted more specific processes D or E, but the more general rule is still better than exporting no rule at all about these proteins, which is the result if the GO hierarchy is not taken into account.

The approach should enrich the annotation of a protein by adding all “is-a” parents of original GO annotations to the protein. For this purpose, the Gene Ontology graph containing the hierarchy and the different relationships between terms had to be imported into the data mining system (named Kraken). The Gene Ontology provides its users with a database containing not only the terms itself but also the relations between them. It is only one single table describing those relationships named GoRelations. It maps child GO ids to parent GO ids and also defines the relation type between them. Figure 3.5 shows a simplified example where all relationships between four terms are contained in three lines of the table described . Both multiple children relations and multiple parent connections are included in a very simple way. This database table has to be converted into a DAG representation in Java. The GO database provides another table, containing all distinct GO terms with their ID, the term itself and its category (molecular function, biological process or cellular component). Similar to the process of making GO annotation accessible for Java classes the database tables are read out and converted into Java objects. GoGraphLoader first retrieves the different terms, creates a GoTermNode for each and stores it in a HashMap using the GO ID as a key. A GoTermNode contains the term itself and a List of parental GO IDs as well as a List of child IDs for each type of relation. These lists are then filled by traversing the GO relations table. For each line a relation is added to the GoTermNode of both GO IDs involved. In the example given, the first line of the table creates a parent for GO:1 and a child for GO:3 by adding the corresponding

GO IDs to their list of "is-a"-children or "is-a"-parents, respectively. Once this is done, the DAG in form of the generated HashMap containing GoTermNodes by their IDs can be serialized and stored for further use.

Child ID	*Parent ID*	*Rel Type*
GO:1	GO:3	is-a
GO:1	GO:4	part-of
GO:2	GO:4	is-a

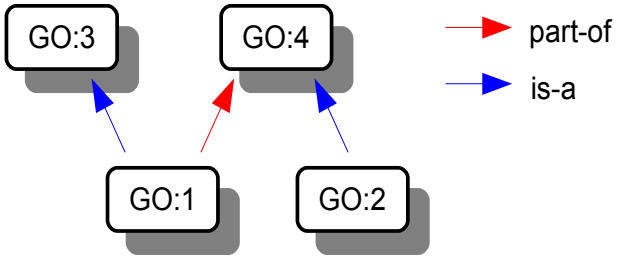

Figure 3.5: Representation of the GO relations in the database. The shown graph is entirely mirrored in the table.

After that, an extended ProteinData object was implemented which is filled with all existing information per protein. Then a simple loop iterates through the GO DAG retrieving all "is-a" parents of the existing GO annotations and adds them to the ExtendedProteinData. This enriched protein can now be used for machine learning. Having "annotated" additional positive GO terms on the proteins provides a more complete GO annotation in terms of combining general and detailed GO terms similar to the taxonomy classification in UniProtKB. Machine learning tools can now find rules about more general GO terms that proteins may have in common.

Negative Enrichment

The other big previously mentioned problem with UniProtKB GO annotation, the missing negative examples, is not yet addressed, and a way to detect such negative examples had to be found. Following the Xanthippe rule exclusion approach (see [6]), a statistical analysis of the data was carried out. To extract as many reliable true negatives as possible from UniProtKB/Swiss-Prot, the rules had to be based on observed rather than predicted data. For this reason, taxonomy classification in proteins was chosen to build exclusion rules for GO terms. An exclusion rule excludes, for example, the GO term "nucleus" for all proteins classified as bacterial proteins. In addition, exclusion rules should be as general as possible. For example, if no bacteria have the GO term "nucleus" then only one rule excluding the term for bacteria should be generated and not a rule for every group of bacteria, for example proteobacteria.

To extract rules like this from UniProtKB/Swiss-Prot, a statistical approach based on the proteins having at least one GO term annotated was chosen. GO terms inferred from electronic annotation are again discarded, i.e. only manually curated GO terms are taken into account. If a GO term is not found in combination with a specific taxonomy classification and the probability for this case is lower than an empirically determined threshold, a rule is exported. The probability of not observing a single GO term *X* in all proteins belonging to a taxonomy *Y* is simply calculated assuming binomial distribution of the number of GO terms observed in all trials (i.e. a uniform probability p over all trials). Formula (12) shows the general formula for calculating the probability of achieving exactly *k* successes in a sequence of *n* independent experiments, each of which succeeds with probability *p* [27]. In terms of an urn experiment, a taxonomy to GO exclusion might be phrased: "When drawing all proteins belonging to a specific taxonomy out of the complete protein set, we do not observe a specific GO term. Is the probability for this case low enough to export an exclusion rule?". Exclusion rules should be extracted if a GO term does not exist within a specific taxonomy, hence in formula (12) *k* is set to zero. *n* is the number of proteins belonging to a taxonomy classification, *p* is the probability of observing a GO term, and *X* is the 'success' of having a specific GO term annotated. The probability of that GO term is estimated by its relative frequency in the complete set of proteins. If *k* equals zero, formula (12) can be converted into formula (13) which is much easier to calculate.

$$P(X=k)=\binom{n}{k}p^{k}(1-p)^{n-k} \quad (12)$$

$$P(X=0)=(1-p)^{n} \quad (13)$$

One drawback of this calculation is the assumption of a binomial distribution, which presumes that the probability *p* does not change during the sequence of *n* experiments. Again in terms of an urn experiment this would mean sampling with replacement. As each protein should only be considered once this is not true in our setting. The hypergeometric distribution describes sampling without replacement and is therefore more suitable in this case. Formula (14) shows the probability of observing *k* times the GO term *X* in a specific taxonomy using the hypergeometric distribution. *M* denotes the frequency of a specific attribute (GO term annotated) in the whole set, *N* is the number of elements in the urn (all proteins having at least one GO term) and *n* is the sample size, which is the frequency of the current taxonomy classification in our setting. Again, *k* is set to zero making

the calculation a bit easier, which is shown in formula (15). Note that *M* in this calculation represents the number of successes in the whole set of elements, which is directly proportional to the probability of success used in the binomial calculation.

$$P(X=k)=\frac{\left(\binom{M}{k}\binom{N-M}{n-k}\right)}{\binom{N}{n}} \quad (14)$$

$$P(X=0)=\frac{\binom{N-M}{n}}{\binom{N}{n}} \quad (15)$$

The main problem using the hypergeometric distribution is its mathematical complexity. Even for formula (15) there are still two binomial coefficients to be calculated which easily result in greater values than a Java "double" can hold (a solution with Java "BigDouble" was implemented for this reason but not used for the reasons described in the following paragraph) or a normal pocket calculator can calculate with. But if the sample size *n* is relatively small in comparison to the size of the complete set of proteins (n/N<0.05), the binomial distribution can be used as an approximation to the hypergeometric distribution. This is visualised in Figure 3.6. Lines represent the probability of observing exactly the same ratio of success to failure in the trial set, which is also found in the whole dataset, or in other words the accuracy of the testing method depending on relation of the number of elements *n* drawn from the dataset to its size *N*. The figure shows two lines for each case out of a set of selected probabilities of success. Dashed lines are calculated by assuming binomial distribution (with replacement). The solid lines result from the same probability calculated with hypergeometric distribution (without replacement). The hypergeometric distribution is much more accurate for greater ratios n/N. The graph approaches a probability of 1 for cases where all elements of the urn were tested, and this is exactly what is expected. At the other end of the range, where n/N becomes very small, the two lines for each probability of success more and more approach each other. The common rule of taking a ratio n/N < 0.05 or 0.1 as a limit for the use of binomial distribution as approximation to the hypergeometric distribution can be confirmed here. Another graph however shows the distance between both calculations in the only case used in this work: where *k* is zero (see Figure 3.7). Depending on the frequency of "successes" in the whole dataset (the frequency of a specific GO term in our case), which directly influences the probability of a

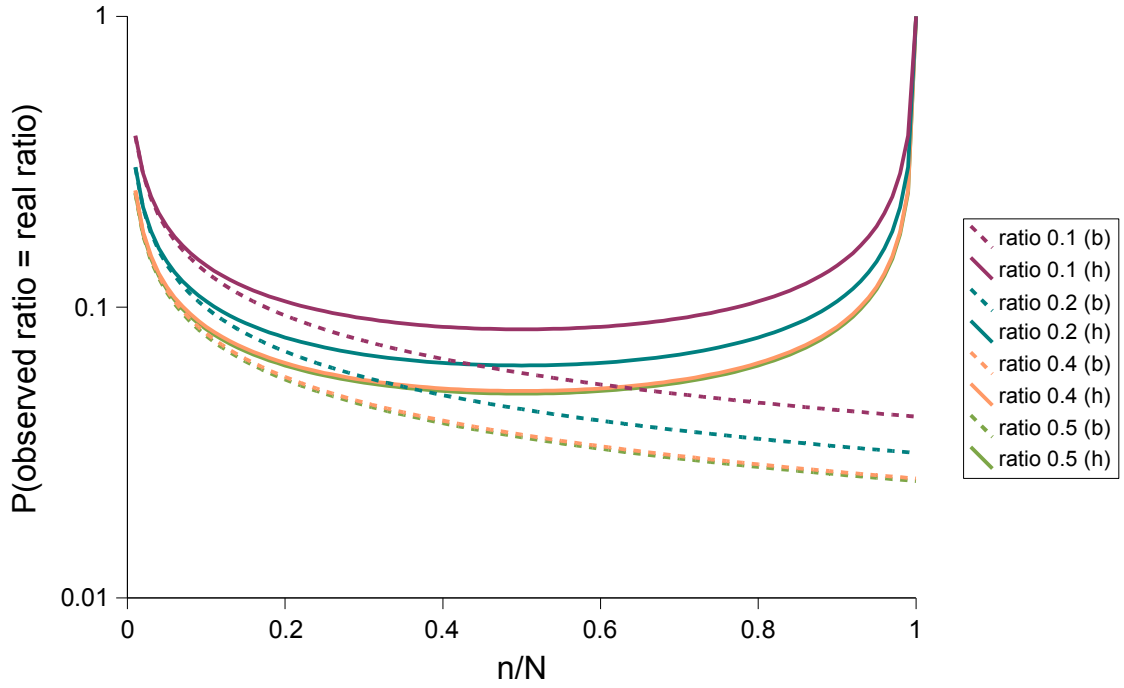

Figure 3.6: Comparison of binomial (b, dashed) and hypergeometric (h, solid) distributions for different probabilities of success. The shown graphs represent the probabilities of observing exactly the same ratio of success to failure as the real ratio is in dependency to n/N. Note that the y-axis is scaled logarithmic which does not change the rough shape of the shown graphs but was done only due to better visualisation.

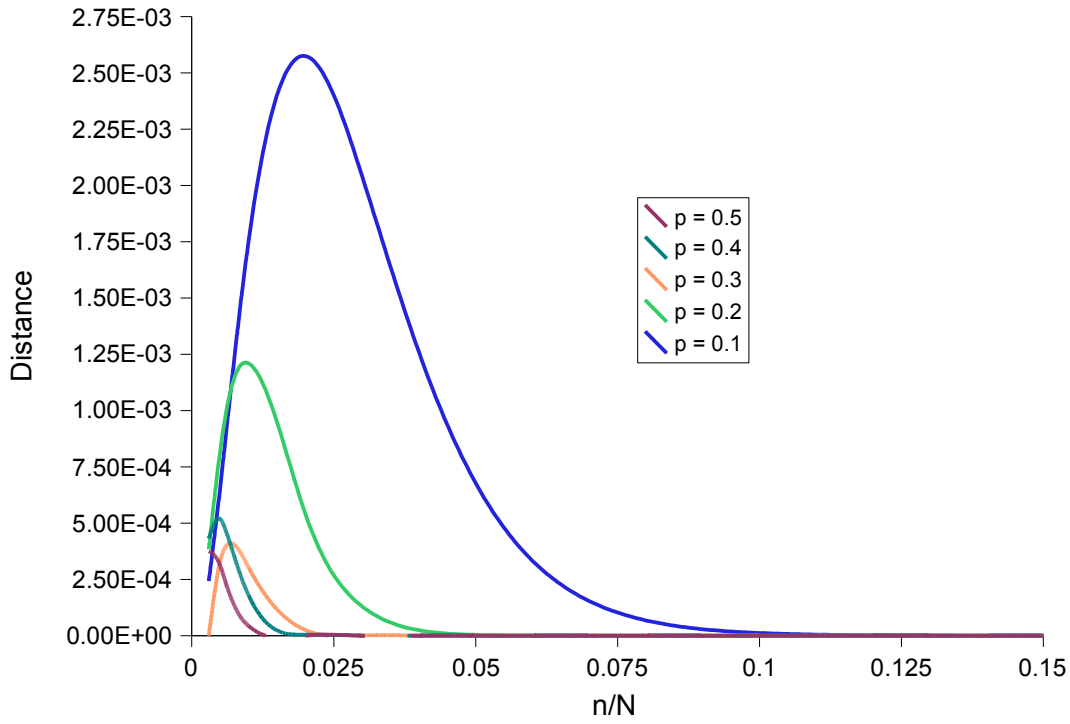

Figure 3.7: Distance between hypergeometric distribution and binomial distribution for several probabilities of success for the case that no success was observed in the trial (k=0)

success, the error rate of the binomial distribution has a peak at varying values of n/N. The smaller p the greater the value n/N of the error peak and the greater the error of the binomial distribution. But note that the maximum error in that graph is only $2.57*10^{-3}$ for p equals 0.1. The next figure (3.8) shows the two distributions and the error rate for a probability of success of only 0.001 where the maximum error is $6.82*10^{-3}$. Note the logarithmic scale of the y-axis. This graph is based on numbers similar to those found in Swiss-Prot, *N* is 35,000 (approximately the number of proteins containing manual GO terms in Swiss-Prot) and *n* is chosen to be 40 which is about the average number of GO term occurrences. Also shown is the threshold of maximum probability to export an exclusion rule at $1*10^{-10}$. This threshold cuts off the probability where the error rate is almost as small as the binomial probability. Even if this means that the error of the binomial calculation is as big as the result itself, the binomial calculation can be used for our purpose. As seen in Figure 3.8, the binomial calculated probability is always higher than the hypergeometric calculated one (for the case that *k* equals zero). In our application, the chance to export a rule about the current GO term and taxon is increasing with lower probabilities for the case where no GO term is found in the

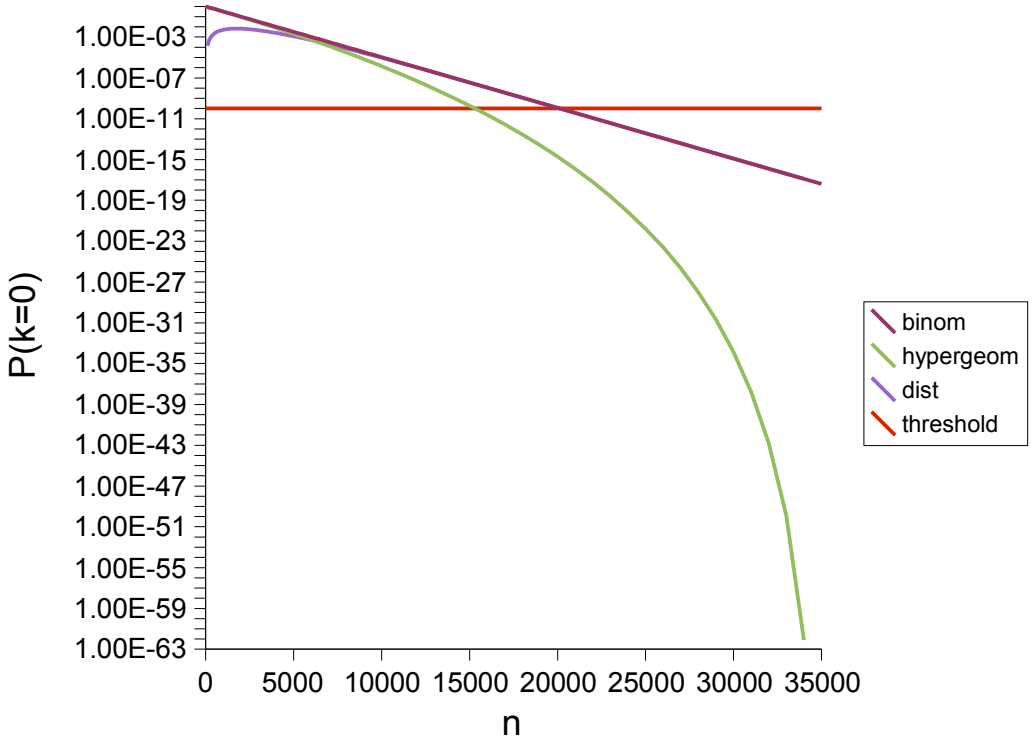

Figure 3.8: Binomial distribution, hypergeometric distribution and the distance between both for zero successes in an example using N=35000 and M=40 (p=0.001) depending on the number of elements drawn from the urn. The maximum error is $6.82*10^{-3}$ but the threshold for maximum probability ($1*10^{-10}$) cuts that off at an error rate very similar to the threshold. Note the logarithmic scale on the y-axis

taxons proteins. So the binomial calculation is the conservative approach and leads to fewer but more reliable exclusions than the hypergeometric distribution. This means, however, that rules which could have been excluded are not, in favour of generating a more reliable set of rules.

The described method provides exclusion rules for different taxonomic classifications. An additional requirement was to ensure that only the most general applicable taxon out of the taxonomic tree is associated with a GO exclusion. This is addressed by sorting all taxons by their frequency in Swiss-Prot under the assumption that the most general taxon is also the most frequent one. Then rules are built and stored in a HashMap containing a List of GO terms for each taxon (the taxon id was chosen to be the key). During the process of rule generation, each new excluding taxon and all its ancestors are looked up in this exclusion map and, only if none of them already excluded the current GO term, a new rule was exported (stored in the map).

The same process is also working for InterPro groups. So exclusion rules look like “InterPro group Y excludes GO term X”. Since InterPro groups do not have a hierarchy like taxons, the lookup during the rule generation step is not necessary.

The exclusion rules generated are then used to provide negative GO associations for the proteins, which are stored just like the additional positive GO terms in the ExtendedProteinData object. If for example a rule is “prokaryote excludes nucleus” and a protein has a taxonomic classification for prokaryotes, then nucleus is added as a negative GO annotation. A further enrichment procedure then takes the GO hierarchy into account and adds additional negatives. This works similarly as the positive enrichment process, only in the opposite direction. For each excluded GO term all more specific “is-a” child terms can also be excluded.

Extending Spearmint

Due to explicit negative GO term associations, the question as to whether a GO term is annotated to a given protein can now result in three different answers. “True” if the GO term is annotated (originally or due to the enrichment process), “False” if it is in the negative GO annotation list and “DontKnow” otherwise.

The Decision Tree algorithm employed by Spearmint can only handle binary results, so a filtering step was implemented to exclude all “DontKnows”. For each currently considered GO term (target), all proteins which returned a “DontKnow”-result for that GO term were filtered out. Thus a set of proteins having either the GO term attached or a negative GO term association was extracted. That set of proteins was then used to train the learning algorithm. Hence the filtering step had to be implemented between the TargetSelectorState and the LearnerState (as seen in Figure 3.2). The

HasGoTermCommand, now returning a Java Integer instead of a boolean, had to be converted back to a boolean command after this filtering step so that the rest of the pipeline could remain unchanged.

4 Results

4.1 Used Measurements

To evaluate the results and compare them, four common measurements were used: accuracy, precision, sensitivity (also called recall) and specificity. These values are all calculated on the base of true positive (TP), false positive (FP), false negative (FN) and true negative (TN) predictions (also see Table 2 on page 23). A definition of them is shown in the following formulas.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (16)$$

$$precision = \frac{TP}{TP + FP} \quad (17)$$

$$sensitivity = recall = \frac{TP}{TP + FN} \quad (18)$$

$$specificity = \frac{TN}{FP + TN} \quad (19)$$

These measures represent the quality of a machine learning algorithm, where accuracy expresses the percentage of all correct predictions out of the sum of all predictions made. Precision means the fraction of all positive predictions that are correct, whereas sensitivity calculates the percentage of positive instances that were also predicted as positive, and specificity expresses the number of negative instances correctly classified as negatives.

4.2 Comparison of the Approaches

All approaches were tested on a small dataset consisting of proteins from two InterPro groups, Kringle proteins (IPR000001) and Connexins (IPR000500). For achieving a comparable set of proteins and high data quality, hypothetical proteins and fragmented proteins were filtered out using the corresponding Swiss-Prot Keywords. To achieve a better set concerning negative associations (for those approaches not based on enriched proteins) records without any manual GO term annotation were filtered out. Thus the size of the sample set of proteins was only 63 and ideal for quick testing and proof of concept. For each run using the described methods, correct and false predictions were counted.

Raw Spearmint and Simple Mapping

In order to show that the existing Spearmint algorithm is not applicable to GO term prediction, first a comparative run of Spearmint on Keywords and then on GO terms was conducted. The result is shown in Table 4. Also included are the results for the Simple Mapping of Keywords to GO terms and of InterPro groups to GO terms. The figures for Spearmint originate from a five-fold Cross Validation on raw protein data out of UniProtKB before any positive or negative enrichment took place. It is easy to recognize the high numbers of true classifications in the Keyword run (417 true positives and 1920 true negatives). 69 false positive predictions of Keywords influence the quality of the classifier more than the 55 false negatives, since negative predictions, i.e. contradictions, will not be transferred to the database in the end. The GO term classifier resulting from the same unchanged algorithm behaves much worse in comparison. Striking is the low number of (true and

	Spearmint		***Simple Mapping***	
	Keywords	***GO terms***	***keyword based***	***IPR based***
TP	417	61	20	21
FP	69	85	6	9
FN	55	73	34	64
TN	1920	1069	460	530

Table 4: True Positive, False Positive, False Negative and True Negative counts for the Spearmint run on Keywords and GO terms as well as for the Simple Mapping based on Keywords and InterPro groups. Training set was the 63 protein sample set before enrichment.

	Spearmint		***Simple Mapping***	
	Keywords	***GO terms***	***keyword based***	***IPR based***
accuracy	0.95	0.88	0.92	0.88
precision	0.86	0.42	0.77	0.70
sensitivity	0.88	0.46	0.37	0.25
specificity	0.97	0.93	0.99	0.98

Table 5: Comparison of accuracy, precision, sensitivity and specificity resulting out of he Spearmint run on Keywords and GO terms as well as for the Simple Mapping based on Keywords and InterPro groups on the small test set before enrichment

false) positive predictions and the ratio of more false positives (85) to less true positives (61). Unfortunately the simple mapping approach didn't improve the over all number of positive predictions. But the ratio of true positives (20 Keyword based, 21 InterPro based) to false positives (6 Keyword based, 9 InterPro based) increased distinctly. The simple mapping also provided a smaller number of negative associations (494 Keyword based and 594 InterPro based) but the false negative numbers (34 and 64) stayed comparably low. Table 5 collects the values for accuracy,

precision, sensitivity and specificity resulting from those absolute counts. Using the rather small set of proteins contained in the two filtered InterPro groups, the prediction result was, as expected, not as significant as if the set had been larger. Nevertheless, the values achieved by pure Spearmint were very good for Keywords (95% accuracy, 86% precision, 88% sensitivity and 97% specificity) whereas the GO term prediction had much worse values, especially for precision (42%) and sensitivity (46%). The simple mapping of Keywords to GO terms lead to a slight improvement over the C4.5 Decision Tree approach on GO terms. The values of accuracy and specificity still lie above 90% but the values for precision and sensitivity differ widely. Whereas the Decision Tree for Keywords showed 86% precision and 88% sensitivity but only 42% precision and 46% sensitivity for GO terms, the simple mapping already shows an almost doubled precision of 77% in comparison with the C4.5 approach, but a slightly lower sensitivity of 37%. That is what follows from a better ratio of true positive classifications to false positive classifications and the resulting decreased number of positive predictions in comparison to negative predictions. The mapping of InterPro groups to GO terms shows a very similar result to the Keyword to GO mapping. An accuracy of 88%, a precision of 70%, a sensitivity of 25% and a specificity of 98% are in general slightly worse than those values for Keyword mapping. Note that for the Simple Mapping algorithm, no Cross Validation was used since the algorithm itself is based on the assumption that ALL proteins are included in the learning step. Hence a run with complete Swiss-Prot would be necessary but could not yet be conducted due to lack of time.

Spearmint on Positive Enriched Proteins

One side product of the enrichment idea was to only positively enrich proteins with parental GO terms. A Spearmint run with those however did not yield convincing results. Accuracy and specificity stayed about the same as with the original proteins (87% and 93%), sensitivity increased slightly to 45% and the precision decreased by 1% to 45%. The number of predictions and contradictions was, as expected, much higher than before (7876 in total) but unfortunately the number of positive classifications increased similarly in false positives (508) and true positives (421).

Negative Enrichment of Proteins

The probabilistic approach using the binomial distribution for the negative search resulted in the original unchanged protein data of whole Swiss-Prot in 719 exclusion rules, which could be pruned to 122 rules due to taxonomy hierarchy. These rules covered 22 different taxonomic classifications and 104 different GO terms. For comparison: The same calculation based on Keywords instead of

GO terms produced an overall amount of 8726 or a pruned set of 1613 rules for 340 different taxonomy classifications which highlights the relatively small number of GO exclusion rules. This again can be explained by lack of GO annotations in UniProt.

Resulting exclusions for GO terms are for example: “Bacteria exclude GO:0005829 (cytosol), GO:0005783 (endoplasmic reticulum), GO:0005739 (mitochondrion), GO:0005634 (nucleus) and GO:0005886 (plasma membrane)”. Eukaryotes exclude some GO terms which have the connotation of bacteria. Those GO terms are marked “sensu Bacteria” for example GO:0030436 (sporulation (sensu Bacteria)). But on the other hand, eukaryotes are also said to exclude “photosynthetic electron transport” or “phycobilisome” which are both involved in photosynthesis and therefore found especially in plants, although there are of course bacteria doing photosynthesis as well. Hence although these rules are not verified by an biologist, some of them look not as reasonable as others. To see whether the principle of negative search could be successful, those rules were nevertheless used for testing purposes. Applied to all manually GO annotated Swiss-Prot entries, approximately 1.4 million negative GO associations were added using these plain exclusion rules. Taking the GO hierarchy into account and enriching all proteins before the negative finder was applied, the new set of rules contained four times more rules. 425 exclusion rules covering 328 different GO terms in 38 taxons could be extracted. Those enriched exclusion rules produced over 3.1 million negative GO terms on Swiss-Prot. But nevertheless in both cases, filtering “DontKnows” out of the learning set for each target lead to a very small group of remaining proteins. To avoid overprediction, only GO terms occurring at a sufficient frequency (3 in the small test set of 63 proteins) were considered as targets. But for all the remaining GO terms no protein having a negative association with that GO term could be found. In this way training sets consisted only of proteins having the target GO term annotated and contained no data to learn negative associations. A test with a greater training set of proteins showed only slightly better results. Almost 90% of all proteins had to be filtered out for each target. The resulting training sets for each target were still too small and therefore excluded from the training. Thus the exclusion rules produced not a single GO term contradiction in the Decision Tree algorithm.

5 Discussion

The results of this work demonstrate that it is challenging to predict GO terms for UniProtKB. Spearmint had difficulties in predicting GO terms, both in the original version and with enriched proteins. Simple Mapping also didn't show convincing improvements and the complementary approach for negative search didn't produce true negative associations. Even though, as seen before, there are some exclusion rules, which are perfectly reasonable. However the investigation of GO annotations in UniProtKB provides valuable insights into the current state of the database.

The biggest problem with GO terms is the lack of annotation. There are only 33,624 entries (13% of UniProtKB/Swiss-Prot) which have at least one GO term (excluding those inferred from electronic annotation). Whereas entries which have at least one Keyword annotated sum up to 246,405 (98% of UniProtKB/Swiss-Prot). Also the distribution of taxonomic classifications among the GO term associated proteins does not reflect the distribution of Organism classifications in UniProtKB/Swiss-Prot. For example, 34,768 (96.7%) of the GO term associated proteins are eukaryotic proteins, whereas in UniProtKB/Swiss-Prot eukaryotes cover only approximately 43% of all proteins. The Gene Ontology consortium was originally founded by research groups only covering eukaryotic organisms, which is the reason that prokaryotic GO annotation started later and is not yet as far on as annotation on eukaryotes. Bacteria are strongly underrepresented among GO term associated proteins with only 1,086 (3%), whereas about 50% of UniProtKB/Swiss-Prot entries are bacterial proteins. Hence since manual GO annotation in Swiss-Prot concentrates on a few model organisms, a future approach will be to also restrict GO term prediction to those organisms. The training set would then only consist of proteins of one specific organism at a time and would hopefully be a better learning base for machine learning algorithms.

One problem in UniProtKB is the true negative association of GO terms. Take a look again at the rules and their coverage of distinct GO terms (104 for the plain exclusion and 328 for the enriched exclusion). In Swiss-Prot, approximately 8,700 distinct GO terms are currently used. This means that for 99% of all GO terms no exclusion rule was found. Considering the exclusion rule set extracted from enriched proteins, still only 4% of all used GO terms were covered. So even if in the example set of proteins each entry was provided with negative associations, these negative associations did not lead to a usable data set for machine learning. The idea of excluding certain GO terms for taxonomic classifications though can be really helpful for future GO annotation. The

existing annotations by GOA could eventually be tested on these exclusion rules and be improved that way. For the manual GO annotation process, the rules can also be useful to make the work easier for curators.

The relatively poor result of Spearmint on enriched proteins is presumably mostly resulting from those still missing real negative examples. The high number of the false positive predictions from Spearmint don't have to be all incorrect. But since Spearmint (without the explicit negative associations) judges non-annotated GO terms as negative examples, even though this is probably in many cases false, such a high number of false positives can be explained. A major part of those false positives could possibly be shifted to true positives or at least to predictions, where it is not possible to say if they are correct or incorrect, if the dataset was more complete.

Other ideas for approaching the problem of automatic GO annotation are to use a probabilistic approach (as Bayesian Nets) instead of a classifier. This approach would for example combine different attributes of a protein and calculate the probability of having a GO term annotated if all those attributes are annotated. That would have the advantage of not needing true negative associations but the disadvantage of not being able to cross validate. This technique together with the exclusions could also be useful for supporting future manual GO curation. Suggestions for specific terms would make it easier for curators to chose correct terms for a protein. This approach applied on existing GO terms only could also help to combine terms from all three domains. The biological process and molecular function of a protein, especially, are often directly connected (e.g.), but GO does not support these relationships between the domains. So a probabilistic approach could extract those connections out of manual GO annotations in UniProtKB and again propose them to curators annotating one of the GO terms involved in such a connection.

A previously mentioned drawback of Decision Trees is that they work best on a balanced training set, where positive and negative examples have similar frequencies. In contrast, Support Vector Machines (SVM), another machine learning algorithm, could handle the great number of outliers and would not need as many true negatives. SVMs are also capable of separating data in more than one dimension so not only linear separation is possible.

A potential problem of the applied approaches is that during the whole work, TrEMBL was assumed to hold negligible information about manual GO annotation. But in fact there are about 37,000 manually GO curated proteins, a comparable absolute number to those in Swiss-Prot. Therefore, it would double the training set and the included knowledge base to include those proteins as well in the training set of all machine learning tools.

What transpires from the conducted database analysis, is that GO annotation in UniProtKB still leaves a lot of room for improvement. Curators could make automatic annotation much easier, if the GO qualifier 'NOT' would be used in a broader way for example[1]. Therefore this qualifier had to be included in the UniProtKB database schema first. In this way, negative associations could be manually curated even if this is not a trivial task of course, keeping the huge number of GO terms in mind. The work of GOA and other curating teams turned out to be very valuable and there is still plenty to do. Maybe the investigated approaches and other ideas could be much more successfully applied to UniProtKB in a few years. But for now, GO data in Swiss-Prot has turned out to be more difficult to use in automatic annotation than expected. To reach reliable predictions on GO terms out of the existing state of the database, a closer examination of the data will be necessary.

1 NOT is actually used by curators in a very small amount of proteins. This is not transferred to UniProtKB though and the usage takes place in a way that is not usable for machine learning. Curators use the qualifier mainly to correct existing annotations which turned out to be false. About 200 proteins have such associations.

6 Appendix

A Example Swiss-Prot Entry

An abridged Swiss-Prot entry in flat file format as an example of a well annotated protein data record. For a better overview headlines were added to the main paragraphs.

```
                        General Information about the entry
ID   128UP_DROME             Reviewed;         368 AA.
AC   P32234; Q9V648;
DT   01-OCT-1993, integrated into UniProtKB/Swiss-Prot.
DT   29-MAR-2005, sequence version 2.
DT   23-JAN-2007, entry version 47.
DE   GTP-binding protein 128up.

                               Origin of the protein
GN   Name=128up; Synonyms=GTP-bp; ORFNames=CG8340;
OS   Drosophila melanogaster (Fruit fly).
OC   Eukaryota; Metazoa; Arthropoda; Hexapoda; Insecta; Pterygota;
OC   Neoptera; Endopterygota; Diptera; Brachycera; Muscomorpha;
OC   Ephydroidea; Drosophilidae; Drosophila.
OX   NCBI_TaxID=7227;

                               Literature References
RN   [1]
RP   NUCLEOTIDE SEQUENCE [GENOMIC DNA].
RC   STRAIN=Oregon-R;
RX   MEDLINE=94166747; PubMed=8121394; DOI=10.1007/BF00281788;
RA   Sommer K.A., Petersen G., Bautz E.K.F.;
RT   "The gene upstream of DmRP128 codes for a novel GTP-binding protein of
RT   Drosophila melanogaster.";
RL   Mol. Gen. Genet. 242:391-398(1994).
RN   [2]
RP   NUCLEOTIDE SEQUENCE [LARGE SCALE GENOMIC DNA].
RC   STRAIN=Berkeley;
RX   MEDLINE=20196006; PubMed=10731132; DOI=10.1126/science.287.5461.2185;
RA   Adams M.D., Celniker S.E., Holt R.A., Evans C.A., Gocayne J.D.,
RA   Amanatides P.G., Scherer S.E., Li P.W., Hoskins R.A., Galle R.F.,
RA   George R.A., Lewis S.E. [...]
RT   "The genome sequence of Drosophila melanogaster.";
RL   Science 287:2185-2195(2000).
RN   [3]
RP   GENOME REANNOTATION.
RX   MEDLINE=22426069; PubMed=12537572;
RA   Misra S., Crosby M.A., Mungall C.J., Matthews B.B., Campbell K.S.,
                                    [...]

                                    Comments
CC   -!- FUNCTION: Deformed (Dfd) is required to activate 1.28up in
CC       maxillary segment cells.
CC   -!- INTERACTION:
CC       Q9VGZ4:CG6325; NbExp=1; IntAct=EBI-163407, EBI-111903;
CC       P25724:nos; NbExp=1; IntAct=EBI-163407, EBI-106556;
CC       Q9V3H7:Sr-CI; NbExp=1; IntAct=EBI-163407, EBI-125222;
CC   -!- TISSUE SPECIFICITY: Expressed in posterior-lateral epidermis of
CC       the maxillary lobe.
CC   -!- DEVELOPMENTAL STAGE: Expressed in embryos and adults.
CC   -!- SIMILARITY: Belongs to the GTP1/OBG family.
```

Database cross-references

```
DR   EMBL; X71866; CAA50701.1; -; Genomic_DNA.
DR   EMBL; AE003823; AAF58591.1; -; Genomic_DNA.
DR   EMBL; AY069810; AAL39955.1; -; mRNA.
DR   PIR; S42582; S42582.
DR   UniGene; Dm.7739; -.
DR   HSSP; P20964; 1LNZ.
DR   IntAct; P32234; -.
DR   GermOnline; CG8340; Drosophila melanogaster.
DR   Ensembl; CG8340; Drosophila melanogaster.
DR   KEGG; dme:Dmel_CG8340; -.
DR   FlyBase; FBgn0010339; 128up.
DR   GO; GO:0005525; F:GTP binding; IDA:FlyBase.
DR   GO; GO:0005515; F:protein binding; IPI:IntAct.
DR   InterPro; IPR006074; GTP1-OBG_dom.
DR   InterPro; IPR006073; GTP1_OBG.
DR   InterPro; IPR002917; MMR_HSR1_GTP_bd.
DR   InterPro; IPR005225; Small_GTP_bd.
DR   InterPro; IPR004095; TGS.
DR   Pfam; PF01926; MMR_HSR1; 1.
DR   Pfam; PF02824; TGS; 1.
DR   PRINTS; PR00326; GTP1OBG.
DR   TIGRFAMs; TIGR00231; small_GTP; 1.
DR   PROSITE; PS00905; GTP1_OBG; 1.
```

Keywords

```
KW   Complete proteome; GTP-binding; Nucleotide-binding.
```

Features

```
FT   CHAIN         1    368       GTP-binding protein 128up.
FT                                /FTId=PRO_0000205430.
FT   NP_BIND      71     78       GTP (By similarity).
FT   NP_BIND     117    121       GTP (By similarity).
FT   NP_BIND     248    251       GTP (By similarity).
FT   CONFLICT      2      2       S -> I (in Ref. 1).
FT   CONFLICT     34     35       KL -> NV (in Ref. 1).
```

Sequence information

```
SQ   SEQUENCE   368 AA;  41132 MW;  5B38B09D0C0A92F2 CRC64;
     MSTILEKISA IESEMARTQK NKATSAHLGL LKAKLAKLRR ELISPKGGGG GTGEAGFEVA
     KTGDARVGFV GFPSVGKSTL LSNLAGVYSE VAAYEFTTLT TVPGCIKYKG AKIQLLDLPG
     IIEGAKDGKG RGRQVIAVAR TCNLIFMVLD CLKPLGHKKL LEHELEGFGI RLNKKPPNIY
     YKRKDKGGIN LNSMVPQSEL DTDLVKTILS EYKIHNADIT LRYDATSDDL IDVIEGNRIY
     IPCIYLLNKI DQISIEELDV IYKIPHCVPI SAHHHWNFDD LLELMWEYLR LQRIYTKPKG
     QLPDYNSPVV LHNERTSIED FCNKLHRSIA KEFKYALVWG SSVKHQPQKV GIEHVLNDED
     VVQIVKKV
//
```

B Example TrEMBL Entry

A typical TrEMBL entry in flat file format as an example of an incompletely annotated record in UniProtKB. For a better overview headlines were added to the main paragraphs.

```
                        General Information about the entry
ID   Q9F0A8_PSESH            Unreviewed;       175 AA.
AC   Q9F0A8;
DT   01-MAR-2001, integrated into UniProtKB/TrEMBL.
DT   01-MAR-2001, sequence version 1.
DT   31-OCT-2006, entry version 20.
DE   HrpD.

                                Origin of the protein
GN   Name=hrpD;
OS   Pseudomonas syringae pv. phaseolicola.
OC   Bacteria; Proteobacteria; Gammaproteobacteria; Pseudomonadales;
OC   Pseudomonadaceae; Pseudomonas.
OX   NCBI_TaxID=319;

                                Literature References
RN   [1]
RP   NUCLEOTIDE SEQUENCE.
RC   STRAIN=1302A;
RX   MEDLINE=21065167; PubMed=11134504; DOI=10.1073/pnas.011265298;
RA   Lee J., Kluesener B., Tsiamis G., Stevens C., Neyt C., Tampakaki A.P.,
RA   Panopoulos N.J., Noeller J., Weiler E.W., Cornelis G.R.,
RA   Mansfield J.W., Nuernberger T.;
RT   "HrpZPsph from the plant pathogen Pseudomonas syringae pv.
RT   phaseolicola binds to lipid bilayers and forms an ion-conducting
RT   pore in vitro.";
RL   Proc. Natl. Acad. Sci. U.S.A. 98:289-294(2001).
RN   [2]
RP   NUCLEOTIDE SEQUENCE.
RC   STRAIN=NPS3121;
RA   Gropp S.J., Guttman D.S.;
RT   "The PCR amplification and characterization of entire Pseudomonas
RT   syringae hrp/hrc clusters.";
RL   Mol. Plant Pathol. 5:137-140(2004).

                              Database cross-references
DR   EMBL; AF268940; AAF99295.1; -; Genomic_DNA.
DR   EMBL; AY530203; AAS20454.1; -; Genomic_DNA.
DR   InterPro; IPR000001; Kringle.
DR   InterPro; IPR013806; Kringle-like.
DR   PROSITE; PS00021; KRINGLE_1; 1.

                                Sequence information
SQ   SEQUENCE   175 AA;  20195 MW;  07FF86135EFABB45 CRC64;
     MELIAEDHWV QWWCNPWQFA HPDWQSRFAL NCGLTLSDCD GLIASRHSVF LQSVGIEPDQ
     PPMPAEPVLR WLALTPLQRE RALDLARRIC FCRNESDGAD GQWCWALTKA LRPGVWLELA
     NEDPRLLLGA WLGPEYWSRL RLAWAPDELP DSPCEAPENK LQTLWQAILW RVTAV
```

C Symbols and Abbreviations

The alphabetic list of Abbreviations contains all used short notations out of the text and also includes the GO evidence codes.

Symbols

$\overline{event}$ ..... Negation, in this case: NOT event

P(event)..... Probability of an event

$P(A|B)$ ..... Conditional probability of event A under the condition of B

Abbreviations

CC.....Swiss-Prot Comment

DAG.....Directed Acyclic Graph

EBI.....European Bioinformatics Institute

EMBL..... European Molecular Biology Laboratory

FN..... False Negative

FP..... False Positive

FT..... Swiss-Prot Feature Table

GO..... Gene Ontology

GOA..... Gene Ontology Annotation at EBI

IC .....Inferred by Curator

ID .....Identifier

IDA..... Inferred from Direct Assay

IEA.....Inferred from Electronic Annotation

IEP.....Inferred from Expression Pattern

IGC..... Inferred from Genomic Context

IGI.....Inferred from Genetic Interaction

IMP..... Inferred from Mutant Phenotype

IPI..... Inferred from Physical Interaction

ISS.....Inferred from Sequence or Structural Similarity

KW..... Swiss-Prot Keyword

NAS..... Non-traceable Author Statement

ND.....No biological Data available

NR..... Not Recorded

RCA.....Inferred from Reviewed Computational Analysis

SP..... Swiss-Prot

TAS.....Traceable Author Statement

TN..... True Negative

TP..... True Positive

TR..... TrEMBL

TrEMBL..... Translation of EMBL nucleotide sequence database

UniProtKB.....Universal Protein Knowledge Base

D Glossary

Descriptions of some terms based on Wilsons Machine Learning Dictionary [28].

Attributes	An attribute is a property of an instance that may be used to determine its classification. For example, when classifying objects into different types in a robotic vision task, the size and shape of an instance may be appropriate attributes. Determining useful attributes that can be reasonably calculated may be a difficult job - for example, what attributes of an arbitrary chess end-game position would you use to decide who can win the game? This particular attribute selection problem has been solved, but with considerable effort and difficulty. Attributes are sometimes also called features.
Binary Tree	Binary Trees are a special kind of tree, in which every node has at most 2 outgoing vertices (out-degree 2).
C4.5	C4.5 is a later version of the ID3 Decision Tree induction algorithm.
Decision Tree	A Decision Tree is a tree in which each non-leaf node is labelled with an attribute or a question of some sort, and in which the branches at that node correspond to the possible values of the attribute, or answers to the question. For example, if the attribute was shape, then there would be branches below that node for the possible values of shape, say square, round and triangular. Leaf nodes are labelled with a class. Decision trees are used for classifying instances - one starts at the root of the tree, and, taking appropriate branches according to the attribute or question asked about at each branch node, one eventually comes to a leaf node. The label on that leaf node is the class for that instance.
Directed Acyclic Graphs	DAGs are graphs with no directed cycles.
Graph	Technically, a (directed) graph is a set V of vertices or nodes, together with a set E of directed edges, which are ordered pairs of vertices. Graphs are widely used in computer science as a modelling tool. A simple example of a graph would be V = {1, 2, 3} and E = {(1,2), (3,1)}, which could be drawn as: 3 -----> 1 -----> 2 A directed cycle in a directed graph is a sequence of edges (v1, v2), (v2, v3), ..., (vn, v1) such that the second vertex of the final edge is the same as the first vertex of the first edge. It is also possible to have undirected graphs, in which the edges are not ordered but rather unordered pairs. Consider the possibility of edges from a node to itself - sometimes these could be useful, sometimes not.
Heuristic	A heuristic is a fancy name for a "rule of thumb" - a rule or approach that doesn't always work or doesn't always produce completely optimal results, but which goes some way towards solving a particularly difficult problem for which no optimal or perfect solution is available.

ID3	A Decision Tree induction algorithm, developed by Quinlan. ID3 stands for "Iterative Dichotomizer (version) 3". Later versions include C4.5 and C5. ID3 chooses a splitting criterion based on information gain. The criterion achieving the greatest gain in information (depending on the purity of resulting classes) will be next to split the dataset.
Instance	An instance is a term used in machine learning particularly with symbolic learning algorithm, to describe a single training item, usually in the form of a description of the item, along with its intended classification. In object oriented programming an instance of a class is a specific (named) object. The class of Dog defines all possible dogs by listing the characteristics that they can have; the object Lassie is one particular dog, with particular versions of the characteristics. A Dog has fur; Lassie has brown-and-white fur. In programmer jargon, the object Lassie is an instance of the Dog class.
Machine Learning	Machine learning is said to occur in a program that can modify some aspect of itself, often referred to as its state, so that on a subsequent execution with the same input, a different (hopefully better) output is produced. See unsupervised learning and supervised learning
Node	A component of a graph or tree.
Over-Fitting	With large complex sets of training patterns, it is likely that some errors may occur. In that case, and particularly in the later parts of the learning process, it is likely that the algorithm will be trained to fit precisely around training patterns that are actually erroneous!
Supervised Learning	Supervised learning is a kind of machine learning where the learning algorithm is provided with a set of inputs for the algorithm along with the corresponding correct outputs, and learning involves the algorithm comparing its current actual output with the correct or target outputs.
Tree Induction Algorithm	This article describes the basic tree induction algorithm used by ID3 and successors. The basic idea is to pick an attribute A with values a1, a2, ..., ar, split the training instances into subsets Sa1, Sa2, ..., Sar consisting of those instances that have the corresponding attribute value. Then if a subset has only instances in a single class, that part of the tree stops with a leaf node labelled with the single class. If not, then the subset is split again, recursively, using a different attribute. This leaves the question of how to choose the best attribute to split on at any branch node. This issue is handled in the article on splitting criterion in ID3.
Tree	Trees are a special kind of directed graph, in which there is a special node, called the root, which has no input vertex (in-degree 0). Every other node has exactly one input vertex (in-degree 1).

Unsupervised Learning	Unsupervised learning signifies a mode of machine learning where the system is not told the "right answer" - for example, it is not trained on pairs consisting of an input and the desired output. Instead the system is given the input patterns and is left to find interesting patterns, regularities, or clusterings among them. To be contrasted to supervised learning, as in ID3, where the system is told the desired or target outputs to associate with each input pattern used for training.

E List of Tables

F List of Figures

7 References

1 Kretschmann E, Fleischmann W, Apweiler R: **Automatic rule generation for protein annotation with the C4.5 data mining algorithm applied on SWISS-PROT.** *Bioinformatics* 2001, **17**:920-926.

2 The UniProt Consortium: **The Universal Protein Resource (UniProt).** *Nucleic Acids Res* 2007, **35**:D193-7.

3 **Uniprot Website** [www.ebi.uniprot.org] (visited in *01/2007*)

4 **Swiss-Prot Release Statistics** [www.expasy.org/sprot/relnotes/relstat.html] (visited in *01/2007*)

5 **Trembl Release Statistics** [www.ebi.ac.uk/swissprot/sptr_stats/index.html] (visited in *01/2007*)

6 Wieser D, Kretschmann E, Apweiler R: **Filtering erroneous protein annotation.** *Bioinformatics* 2004, **20 Suppl 1**:I342-I347.

7 Kleen M: **Sequence-based Feature Prediction on Proteins**. *Master thesis.* University of Applied Sciences Weihenstephan. 2006.

8 **The Gene Ontology Website** [www.geneontology.org] (visited in *01/2007*)

9 Apweiler R, Attwood TK, Bairoch A et al.: **InterPro--an integrated documentation resource for protein families, domains and functional sites.** *Bioinformatics* 2000, **16**:1145-1150.

10 Camon E, Magrane M, Barrell D et al.: **The Gene Ontology Annotation (GOA) project: implementation of GO in SWISS-PROT, TrEMBL, and InterPro.** *Genome Res* 2003, **13**:662-672.

11 Zehetner G: **OntoBlast function: From sequence similarities directly to potential functional annotations by ontology terms.** *Nucleic Acids Res* 2003, **31**:3799-3803.

12 Hennig S, Groth D, Lehrach H: **Automated Gene Ontology annotation for anonymous sequence data.** *Nucleic Acids Res* 2003, **31**:3712-3715.

13 Vinayagam A, Koenig R, Moormann J et al.: **Applying Support Vector Machines for Gene Ontology based gene function prediction.** *BMC Bioinformatics* 2004, **5**:116.

14 Hvidsten TR, Komorowski J, Sandvik AK et al.: **Predicting gene function from gene expressions and ontologies.** *Pac Symp Biocomput* 2001, :299-310.

15 Barutcuoglu Z, Schapire RE, Troyanskaya OG: **Hierarchical multi-label prediction of gene function.** *Bioinformatics* 2006, **22**:830-836.

16 King OD, Foulger RE, Dwight SS et al.: **Predicting gene function from patterns of annotation.** *Genome Res* 2003, **13**:896-904.

17 Gruber TR: **Toward principles for the design of ontologies used for knowledge sharing.** *Int. J. Hum.-Comput. Stud* 1995, **43**:907-928.

18 The Gene Ontology Consortium: **Creating the gene ontology resource: design and implementation.** *Genome Res* 2001, **11**:1425-1433.

19 **Amigo! A Gene Ontology Browser** [www.godatabase.org] (visited in *02/2007*)

20 Batista CVF, del Pozo L, Zamudio FZ et al.: **Proteomics of the venom from the Amazonian scorpion Tityus cambridgei and the role of prolines on mass spectrometry analysis of toxins.** *J Chromatogr B Analyt Technol Biomed Life Sci* 2004, **803**:55-66.

21 Cherry JM, Adler C, Ball C et al.: **SGD: Saccharomyces Genome Database..** *Nucleic Acids Res* 1998, **26**:73-79.

22 Quinlan J: **Induction of decision trees.** *Machine Learning* 1986, **1**:81-106.

23 Quinlan JR: **C4.5 Programs for machine learning.** The Morgan Kaufmann Series in Machine Learning; 1993.

24 Witten IH, Frank E: **Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations.** MorganKaufmann Publishers; 2000.

25 Mulder NJ, Apweiler R, Attwood TK et al.: **New developments in the InterPro database.** *Nucleic Acids Res* 2007, **35**:D224-8.

26 Gamma E, Helm R, Johnson R et al.: **Design Patterns Elements of Reusable Object-Oriented Software.** Addison-Wesley Longman Publishing Co., Inc.; 1995.

27 **Binomial Distribution In Wikipedia** [http://en.wikipedia.org/wiki/Binomial_distribution] (visited in *12/2006*)

28 **The Machine Learning Dictionary** [www.cse.unsw.edu.au/~billw/mldict.html] (visited in *03/2007*)