AUTOMATIC FUNCTIONAL ANNOTATION OF PROKARYOTES

by

Philip Goetz

A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Bioinformatics

Committee:

	Dr. Iosif Vaisman, Thesis Director
	Dr. Ancha Baranova, Committee Member
	Dr. Don Seto, Committee Member
	Dr. James Willett, Director, School of Systems Biology
	Dr. Timothy L. Born, Associate Dean for Student and Academic Affairs, College of Science
	Dr. Vikas Chandhoke, Dean, College of Science

Date: Dec. 6, 2011

Fall Semester 2011
George Mason University
Fairfax, VA

Automatic Functional Annotation of Prokaryotes

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Philip Goetz
Doctor of Science
State University of New York at Buffalo, 1997

Director: Iosif Vaisman, Professor
Department of Bioinformatics

Fall Semester 2011
George Mason University
Fairfax, VA

Dedication

To my nieces and nephews, who had to play the monster game on their own while their uncle was writing this thesis.

Acknowledgements

The following people also contributed to the development and testing of AutoAnnotate:

- Tanja Davidsen, Granger Sutton, and Lauren Brinkac: Project managers
- Ramana Madupu: HMP liason
- Tanja Davidsen, Ramana Madupu, Scott Durkin, Bob Dodson, Scott Durkins: Annotators, specified requirements, testing and production use
- Bill Nelson, Kevin Galens: Wrote the prior version of autoAnnotate
- Alex Richter: Wrote code to parse BLAST output; perpetual Perl consultant
- Bob Montgomery: Revised the comparison tool

Table of Contents

Page

List of Tables

Table **Page**

Abstract

AUTOMATED CONSTRUCTION OF A RANKING SYSTEM FOR AUTOMATIC FUNCTIONAL GENE ANNOTATION

Philip Goetz, MS

George Mason University, 2011

Thesis Director: Dr. Iosif Vaisman

One key method of automatic functional annotation of a prokaryote gene is finding BLAST hits to the gene in question that have functional annotations, choosing the best single hit, and copying the annotation from that hit if it is of sufficient quality as measured by a p-value or other criterion. In the JCVI prokaryote automatic functional annotation system, the best hit is chosen by looking up categories in a manually-constructed table stating how reliable the annotation is depending on who made the annotation, what percent identity the BLAST hit had, and what percentage of the query gene and the hit gene were involved in the match.

Constructing this table is labor-intensive; and humans are incapable of processing enough data to construct it correctly. I therefore reduced the data requirements by breaking the table into orthogonal components; and I developed an iterative method to

minimize the least-squares error of the table on a training set. I also constructed a validation set of 50,000 manually-annotated proteins from JCVI data, and developed a protein name thesaurus and ontology to make it possible to tell when two names meant the same thing, or when one name was a more-specific refinement of another name. Training on 9/10 of the validation set, and testing on the held-out 1/10, showed an improvement in accuracy from 61.5% to 72.4%.

Chapter 1: Overview

1.1. Functional Annotation

Annotation is divided into two parts. *Structural annotation* means noting where genome features begin and end, and other features such as frameshifts. (Structural annotation is more complicated for eukaryotes; I consider only annotation of prokaryotes.) *Functional annotation* means describing what a genome feature does.

In this thesis, the only type of genome feature we will talk about is genes. Gene features also include RNAs; notably, rRNAs, tRNAs, and various regulatory RNAs. (Viral inserts and pseudogenes are generally annotated as if they were genes.)

Annotations must fit into the Genbank format, since otherwise they are lost when the data are uploaded to Genbank. Genbank does not use any controlled vocabulary other than EC number; and EC numbers are often not very specific. So while you can write programs that guess at a protein's cellular localization, expression pattern, operon membership, melting temperature, or other properties, you can't store that information in a way so that it will be found again. This is because Genbank, and annotation in general, still has a "one-gene" culture, where the expectation is that the users of annotation are scientists investigating a single gene, who look up that gene's annotation and read it.

Even today, you can find papers on functional annotation that never describe what the annotations are (e.g, Aziz et al. 2008). In such cases, you can usually presume that the annotations are simply descriptions of function in English.

The Genbank format accepts the following fields relevant to functional annotation of genes:

- Gene name
- Secondary gene name
- Standard name (unclear what this means)
- Protein product name
- Function
- EC number
- Definition: This is usually taken as a "common name" plus comments.
- Keywords
- Database accession numbers (dbxref; used to store GO terms)
- Notes

The situation was greatly improved with the adoption of the Gene Ontology (Gene Ontology Consortium 2000). Functional annotations can now include assignments of GO IDs. Usually, these are from the biological process and molecular function hierarchies; less work has been done using the cellular component hierarchy. The great advantage of the GO IDs is that different software and different genome sequencing centers can assign the same GO IDs. This means there is some hope of noting whether

different annotations agree or disagree; of summing up evidence from multiple sources; and of searching for genes with particular annotations.

Genbank can accept GO ID annotations (using the /db_xref keyword), but JCVI does not submit GO IDs to Genbank, because Genbank throws away the GO evidence codes (http://www.geneontology.org/GO.evidence.shtml) and the comments on experimental parameters. Hence, although we have a standard ontology, it still is not very useful for finding genes with particular annotations.

Genbank does not accept any statement about confidence level in an annotation. This has had a bad effect on annotation, and on the culture of annotation; the annotators at JCVI have spent over a decade trying to make annotations only when they were about 98% certain of the truth. TIGR (and, by extension, JCVI) have achieved a reputation as being a genome sequencing center with very high standards of evidence needed for annotation, and they pride themselves on that. But that results in a higher false negative rate, which is a problem for some users. Because Genbank does not accept a confidence level for an annotation, JCVI has never computed one, beyond deciding how specific to make the protein name. But different end-users have different costs for false positives, and so should be able to make their own decisions about what level of confidence they require in an annotation (King et al. 2001).

The limited number of evidence types specifiable using the GO Consortium's evidence codes (http://www.geneontology.org/GO.evidence.shtml) is also a problem. These codes were developed for curated annotations; there is only one code, IEA, used for all automated annotation. Thus, there is no way to distinguish between annotations

produced from reliable, curated sources; and annotations produced from unreliable, uncurated sources. Despite the fact that the go Consortium web page explicitly says, in large red letters, that evidence codes are not to be used to indicate reliability of annotations, that is what people mainly use them for; and it is common practice to throw out any annotations using the evidence code IEA due to this inability to distinguish good from bad automatic annotation.

Originally, annotations were made by humans for humans. Today, annotations are made by computers for humans. Inevitably, annotations will be made by computers for computers (Slater et al. 2008). That is, whereas today we have an investigator look up the annotation for a single gene, in the future we will more and more see computer programs retrieving the annotations for many genes. It is wrong to hide information from a computer program by declining to say anything about a protein, rather than saying that you have an idea what the protein is, but are only 80% confident. One of the goals of this work is to provide a way for the JCVI annotation pipeline to produce calibrated probabilities for its annotations, in preparation for the day when there is a demand for them and a way to record them.

Chapter 2: Literature Review

Automatic functional annotation systems can be usefully categorized according to the data sources used; the method used for classifications; whether classification uses machine learning or not; and how to combine multiple evidence sources.

Most papers on automated annotation give validation results. However, it is impossible to compare results across any two papers, because they are measured using different statistics (precision vs. recall and sensitivity versus specificity are popular), annotating different genes, using different evidence types and different annotation databases to transfer annotation from, producing different output classes, using different cutoffs for classification. It would probably be preferable to express results in terms of reduction of output distribution entropy. No one has done this to date. Papers that compare several different automatic annotation under the same conditions include (Lee et al. 2006, Chua et al. 2007, and Peña-Castillo et al. 2008).

2.1. Data sources

The following list accounts for almost all, if not all, data used in automated functional annotation:

- Sequence: Gene nucleotide or protein sequences can be compared via BLAST or multiple alignments; or they can be matched using regular expressions, hidden Markov models, or machine-learning classifiers.
- Protein interactions: This is a variety of predictions or experimental evidence that two proteins either associate with each other, are members of the same protein complex, or interact with the same small molecules or pathogens. Most of this data is experimental; predictions are mainly restricted to predicted binding sites through which proteins might interact.
- Co-occurrence: This includes co-occurrence in text (usually PubMed abstracts), and gene co-expression data.
- Genome context: This refers both to *synteny,* when genes that work together to fulfil one function are located next to each other on the genome; and to expectations about whether the larger pathways that a protein function takes part in occur or do not occur in this organism, and which particular genes in those pathways have not yet been identified.
- Structure: Using 3D structure to predict function, in much the way sequence is used. (When 3D structure is predicted from sequence, we treat this as being sequence-based.)

Some data is *simple* (not a standard term), meaning that it is taken directly from the world using an unambiguous procedure. This includes sequence data, textual co-occurrence, gene co-expression, and other experimental data. *Complex* data is a term I shall use for data that has been aggregated from simple data, to such a degree that the

aggregations themselves are controversial and need to be validated or curated. For example, the construction of protein families (e.g., COGs (Tatusov et al. 2000) or TIGRFAMs (Haft et al. 2003) or expert-system rules (e.g., RuleBase (Biswas et al. 2002) or BrainGrab) produces complex data.

For simple evidence, reliability is a matter of the accuracy of the measurements (for experimental data), and the strength of the evidence type. Gene co-expression data or protein interaction data are not reliable enough to base an annotation on, even if the data were perfect.

For complex evidence, including previous annotations, reliability is primarily a function of the degree of curation. *Curated* data has been either experimentally verified, or each piece of data has been inspected by a human. This includes data that is machine-generated, but later inspected by humans. For example, NCBI COGs are generated automatically using BLAST, and then checked and refined by human curators. *Uncurated* data was machine-generated and may never have been checked by a human. This includes previous automatically-assigned annotations. EMBL divides their database of annotations into curated (Swiss-Prot) and uncurated (TrEMBL) annotations.

Transitive annotation is when a new gene is given an annotation based on sequence similarity to a gene with annotation. JCVI does not allow transitive annotation from uncurated annotation.

Validation and curation are operationally similar but conceptually different. *Curation* means the goal is to verify the correctness of the particular annotation being

made, while *validation* means performing curation in order to verify the correctness of the process that assigned the annotation. For instance, a TIGRFAM specifies a process for determining whether a protein sequence matches that TIGRFAM; and when a curator constructs a TIGRFAM, the curator adjusts its cutoff levels and phylogenetic scope until it gives no false positives on JCVI's genome database. This is validation of the TIGRFAM rather than curation of a particular annotation. It is a special case of validating an automatic annotation method against a database, and then trusting its output in the future.

Different data sources are curated to different degrees. SwissProt annotations and TIGRFAMs are considered by many annotators to be reliable enough to make an annotation based on a single match to a gene with a SwissProt annotation or a TIGRFAM family (see Louie et al. 2008 for a validation competition including SwissProt and TIGRFAMs).

Automatic annotation faces a major difficulty in the near future. More and more genomes are being sequenced, and fewer and fewer curated annotations are being produced. Funding agencies would generally rather pay to have hundreds of new genomes sequenced and run through an automatic annotation system, than to have five genomes sequenced and annotated by hand. But automatic annotation relies on identifying previously-annotated genes and transferring their annotation to the new gene. There are no reliable de-novo predictions of protein function; even if the technical problems were solved, it would be very difficult to translate a de-novo predicted function into English that a biologist would recognize. Doubling the number of curated

annotations would improve automatic annotation more than all the fancy algorithms in the world.

2.1.1 Sequence

Almost all evidence sources used for functional annotation are sequence-based. The primary sequence-based tool is BLAST, which is used to find similar genes. If a sequence matches an already-annotated sequence closely enough, it is presumed that the two sequences have the same function. Sometimes there are additional requirements. The JCVI automatic annotation will not transfer annotation unless the original annotation was experimentally characterized.

BLAST is the primary tool for functional annotation. The precision-recall graphs in figure 6 of (Chua et al. 2007) show that BLAST alone accounts for perhaps 95% of the area under the precision-recall curve for predicting molecular function when 10 different evidence types were used. BLAST is less dominant, though still dominant, for biological process and cellular component. It is therefore crucial to be aware, when comparing the performance of different annotation systems, whether or not they are using BLAST. (Deng et al. 2004) do not use BLAST in their system, and so their not-very-impressive-sounding 57% precision at 87% recall with only 13 output categories should not be compared directly to the results of a system using BLAST.

Hidden Markov models (HMMs) are also sequence-based; for example, TIGRfams (Haft et al. 2003), Pfams (Finn et al. 2008), and PANTHER (Mi et al. 2005, 2007). Regular expressions can be matched to sequences, as PROSITE does (Sigrist et

al. 2002). There are also a host of special-purpose sequence-based methods for guessing at particular properties of proteins; the most commonly-used are SignalP (for predicting signal peptide cleavage sites; Emanuelsson et al. 2007), TargetP (Emanuelsson et al. 2000) and Psort (Gardy et al. 2005) (for predicting subcellular localization based on sorting signals), and TMHMM (for predicting transmembrane helices; Krogh et al. 2001).

All protein families used for annotation are sequence-based. That seems like an outrageous claim, given that there exist well-developed non-sequence-based protein families in databases such as CATH, SCOP, and FSSP. However, I am not aware of anyone using any of these for functional annotation. Besides the already-mentioned Pfams, TIGRFAMs, and PANTHER, there are also FIGfams (Overbeek et al. 2005) and NCBI COGs (Tatusov et al. 2000). The Ensembl annotation pipeline (Curwen et al. 2004; but this reference doesn't describe this fact, which was relayed at a recent conference) relies heavily on combining evidence from these, and other, protein families.

The predominance of sequence-based data is a major problem for automated annotation. Different types of sequence-based data are never orthogonal. Many annotation systems are built using a variety of sequence-based data; the marginal improvement gained by adding each new type of sequence-based data is slim, because it is all correlated. Often, adding a new sequence-based data source gives no improvement at all (Chua et al. 2007).

2.1.2 Protein interactions

There is a large literature on interaction-based (also called network-based) annotation; see (Sharan et al. 2007) for a review. However, they are seldom used in functional annotation, because the evidence provided is very weak; and, as mentioned above, the annotation community would rather say nothing about a gene than assert a function with only 80% confidence. There is also a tendency for interaction-based function prediction to be of interest to a more mathy community that prefers approaches that are too computationally-intensive to actually produce annotations, as witnessed by the fact that (Sharan et al. 2007) cites over 100 references, yet has not a single reference to an operating functional annotation system; and only a handful of his 100+ citations have been cited in papers by people who actually do functional annotation. (Chua et al. 2007 is the source through which I originally found the aforementioned handful of papers cited by Sharan et al., and they note that most of them – Deng et al. 2004, Lanckriet et al. 2004, and Lee et al. 2006 – use methods that are too computationally intensive to be practical.)

STRING (von Mering et al. 2007) and STITCH (Kuhn et al. 2008), both from EMBL, combine sets of non-sequence based evidence, including co-occurrence in Pubmed abstracts, microarray co-expression, gene fusions, and protein-protein interaction data. (I am not aware of any automatic annotation systems that use STRING; we hope to use it at JCVI. EMBL's automatic Ensembl annotation pipeline does not use STRING; an Ensembl trainer I spoke to was not even aware of its existence.)

Mass spectrometry can give information about post-translational modifications that indicate cellular localization or turnover rate. However, to date, the only use of mass spectrometry in annotation has been for structural annotation, where it can be very useful for determining the correct start and stop sites for a gene, and for detecting frame shifts (Ishino et al. 2007).

2.1.3 Co-occurrence

One form of co-occurrence is co-citation in PubMed abstracts. One co-citation is very weak evidence; but according to automated validation using the KEGG pathways as a gold standard (von Mering et al. 2007), a large number of co-citations can sometimes give a 90% confidence that two proteins interact. Interaction, however, is no guarantee of sharing function. As mentioned above, STRING and STITCH use co-occurrence information.

Another form of co-occurrence is gene expression, as determined by microarrays. STRING provides co-expression data. Gene expression was used by (Pavlidis et al. 2002), as well as by all entrants in the functional annotation competitions described in (Chua et al. 2007) and (Peña-Castillo et al. 2008). By itself, it is not very useful evidence; (Chua et al. 2007) demonstrated that in order to have more than 20% precision in predicting GO IDs from co-expression data alone, it is necessary to reduce recall to less than 5% for biological process, and 10% for cellular component and molecular function. For these latter two categories, however, precision can be as high as 85%, if

recall is reduced to about 3%. Predicting function for 3% of proteins of unknown function would be a worthwhile accomplishment.

2.1.4 Genome context

Genome context, usually synteny, is a powerful tool frequently used by human annotators. The RAST annotation pipeline, which has been used on 350 genomes (Aziz et al. 2008), uses genome "subsystems" (Overbeek et al. 2005), which are pathways or systems known to occur in several genomes. STRING (von Mering et al. 2007) also uses synteny as one of its data sources. If an unidentified gene is surrounded by other genes that are homologs to genes in one of these subsystems, its homology to one of the other genes in the subsystem is strongly indicated. JCVI uses a similar system called Genome Properties (Selengut et al. 2007) for curated annotation. Xanthippe, used by EMBL, also uses genome context: It uses machine learning of decision trees to remove annotations that are unlikely due to taxonomy (Wieser et al. 2004).

FIGENIX (Gouret et al. 2005) uses phylogeny to supplement BLAST to distinguish orthologs from paralogs. TIGRFAMs also use phylogeny to restrict hits to within a certain phylogenetic scope of the species used to construct the TIGRFAM (Haft et al. 2003). It is surprising how little phylogeny is used in automatic functional annotation, since a major source (likely the major source) of errors in functional annotation is the assignment of function based on a BLAST hit to a paralog.

2.1.5 Structure

Structure-based methods are rarely used, other than structure inferred from sequence (e.g., Warmr, in King et al. 2001). Structural homology is more predictive of function than sequence homology, but most newly-sequence genomes have no known protein structures. Because of the high cost of determining a protein's structure, few structures are known (fewer than 10,000 unique structures are in PDB); and it used to be a process reserved mainly for proteins of known function. The Protein Structure Initiative, however, solves the structures mostly of proteins of unknown function (Chandonia & Brenner 2006). As the number of known structures for proteins of unknown function increases, use of structure should increase. (Ward et al. 2008) used 3D motif-matching to achieve in predicting the EC numbers of 1314 proteins with 39% recall and 92% precision.

2.1.6 Enzymatic screening

(Kuznetsova et al. 2005) appears to be the only publication so far describing the use of high-throughput enzymatic microarray screening to provide functional annotations for uncharacterized proteins. The authors devised general enzymatic assays for different broad classes of protein function. Other groups are beginning similar projects, including the Yeast Integrative Biology Projects at the Ontario Genomics Institute, and the Biosciences Division of Argonne National Laboratories. Neither have any publications listed on their websites as of November 2011.

2.2. Classifiers and machine learning

Other than gene names and descriptions, annotations are usually classifications (one or more labels from a finite set of labels). These include EC number (Haft et al. 2003, Jensen et al. 2003, C. Yu et al. 2008), GO IDs (Haft et al. 2003, Jensen et al. 2003, Eisner et al. 2005, C. Yu et al. 2008) or GO slims (Biswas et al. 2002), TIGR role / Riley classification (Genequiz Andrade et al. 1999, Warmr King et al. 2001, Haft et al. 2003, Jensen et al. 2003), subcellular location (Emanuelsson et al. 2000, Biswas et al. 2002, Gardy et al. 2005), MIPS functional class (Deng et al. 2002, Pavlidis et al. 2002), and Swiss-Prot keyword (Biswas et al. 2002, Schroeder et al. 2002, Gattiker et al. 2003).

After BLAST match to an annotated gene, one can simply transfer the annotations from one gene to another. Other types of evidence, however, are not so conclusive, and so different pieces of evidence, and different types of evidence, must be combined by a classifier to guess at an output category.

Rules can be built by hand to produce classifications. Alternately, many machine-learning techniques can be used to automatically construct classifiers from gold-standard curated annotations. (Machine-learning techniques can also produce rules, but this is not usually the case. Likewise, one could create a decision tree, or even a neural network, by hand; this is not usually done.)

A single classifier can be used for each output type to combine all data from all evidence types and produce an output class. We will revisit rule-based systems and decision trees in the next section because that is how they are usually used, whether or

not they are produced via machine learning. In some cases, however, classifiers are used to produce intermediate outputs, which will later be combined to produce the final annotations. For example, PSORTb (Gardy et al. 2005) uses 9 support vector machines to produce yes/no decisions for each of its 9 output classes; and uses a completely different approach – a Bayesian network – to combine the outputs of these 9 classifiers into a final decision. In this section, I will cover only machine learning of classifiers, and not data integration using classifiers.

2.2.1 Rule-based systems

RuleBase generates conjunctive rules from InterPro accession hits (Fleischmann et al. 1999, Biswas et al. 2002). (G. Yu 2004) describes itself as a rule-based system that automatically construct rules, but it is not clear whether these "rules" have variable bindings or conjunctions, or are just a big lookup table.

Learning cutoffs that determine when a match to a protein motif or family should apply, as in (C. Yu et al. 2008), is a more restricted form of rule learning.

2.2.2 Decision trees

Decision trees are similar to rule-based systems; but instead of having a collection of rules, any one of which can make a classification, they are a tree of (usually binary) decisions. A path from the root of the tree to a leaf is analogous to a rule in a rule-based system. Whereas rules in rule-based systems are independent of each other, and the construction of one rule does not have much effect on other rules; a decision tree is constructed by using information theory to place the most informative test at each branch

point in the tree; changing the test used at one branch point would change all of the tests underneath it.

Decision trees work with input data that comes in discrete classes. Continuous data (e.g., BLAST e-values) must be divided into bins. This is both their greatest advantage (different evidence types don't need to be normalized in order to be used together), and their greatest disadvantage (much information is lost during the binning). They are used in Warmr (King et al. 2001), Spearmint (Kretschmann et al. 2001), Xanthippe (Wieser et al. 2004), and in (Schroeder et al. 2002).

2.2.3 Neural networks

Neural networks are used in (Schroeder et al. 2002), who found them to be 99.23% accurate, but no better than decision trees, and less easy to understand the output from. The high accuracy claimed may be attributed to the fact that they restricted themselves to transitive annotation within the Mycoplasmataceae family and excluded hypothetical proteins. It is difficult to know whether anything can be learned from performance on such an easy task. (Jensen et al. 2003) found a recall of 60-80% with 10% false positives.

2.2.4 Support vector machines (SVMs)

SVMs project input vectors into a higher dimensional space, and then find a plane that best discriminates between two classes in that space. They are used in (Pavlidis et al. 2002, Jensen et al. 2003, Lanckriet et al. 2004, Gardy et al. 2005, Eisner et al. 2005). They were shown to be very effective compared to other classification methods in (Lee et

al. 2006, Chua et al. 2007, and Peña-Castillo et al. 2008). Computationally, they are very demanding. Because they perform only binary classifications, what is typically done is to train a separate yes/no SVM for each output class, and then combine the outputs of the SVMs. This increases the computational burden.

2.3. Combining different evidence types

The data sources listed above provide different levels of coverage, different levels of certainty to infer similar properties across a link that depend on the type of property you are trying to infer, and different degrees of independence from other data sources (Chua et al. 2007, Peña-Castillo et al. 2008). For example, sequence-based methods can infer similar molecular function with high confidence from a single association, but provide less certainty when inferring biological process (Chua et al. 2007). Co-expression provides information that is of little value for inferring similar molecular function by itself (Chua et al. 2007), but may be valuable in supplementing sequence-based information because it is highly non-correlated with it. Experimental validation shows that different experimental protein-protein interaction paradigms provide largely independent data on interactions between different types of proteins (H. Yu et al. 2008).

It is thus important to have a statistical method for combining all of this information, both for summarizing to users, and to use in automated functional annotation; and to validate this method for every data source using experimental data.

2.3.1 Best hit

"Best hit" means that each evidence type provides a confidence level with each piece of data, and for each gene, you find the piece of data of any evidence type that has the highest confidence level, and use it, and throw the rest out. You may take a different best hit for each type of output you produce. JCVI's prokaryotic automatic annotation system does this. The main advantage is that you don't have to do any training to set weights (as in voting) or to map confidence scores into probabilities (as with naïve Bayes).

2.3.2 Decision tree

Decision trees, described above, are typically used to produce a final output when they are used at all, since it is always easy to throw more data into a decision tree. If they are given many different data inputs, they can have the problem that the constructed tree comes to a leaf node, at which all training samples are classified correctly, before it has used all of its input types. This can cause it to throw away large amounts of information.

2.3.3 Voting

In voting, each evidence, or each evidence type, produces an output class, and some algorithm determines the final output class from those votes. The simplest scheme is majority voting. Precision can be increased at the expense of recall, by requiring at least two votes to produce an annotation. (This is, of course, a bad thing to do; better is to always produce an annotation, but also record the confidence of that annotation.

However, if evidence types are not validated (and they often are not), producing confidences is not very meaningful.)

Different evidence can be weighted differently from validation data. (Pavlidis et al. 2002) weighted voting by validation. Warmr (King et al. 2001) and GeneMANIA (Mostafavi et al. 2008) used validation with regression to weigh votes. Regression prevents correlated evidence types from exerting too much combined influence. (Eisner et al. 2005) found weighted voting was no better than majority vote when combining PFams, Proteome Analyst features, HMMs, and BLAST hits. Regression-weighted voting might have improved that.

2.3.4 Naïve Bayes

Naïve Bayes computes a probability for each annotation by assuming that all of the evidence sources are statistically independent. STRING (von Mering et al. 2007) uses naïve Bayes. PIPA (C. Yu et al. 2008) showed Naïve Bayes to be superior to taking the best hit, although they did not use BLAST as a simple data source, but to construct protein family profiles. Two recent comparisons of different annotation systems gave opposing results about the effectiveness of Naïve Bayes: (Chua et al. 2007) indicates that naïve Bayes performed as well as any more sophisticated integration techniques; while (Peña-Castillo et al. 2008) indicated that naïve Bayes performed much worse than all other integration techniques. The difference may be attributed to the fact that the former used BLAST, which so dominated the results that the addition of other data sources

hardly mattered; while the latter did not use BLAST as evidence, and so how data sources were integrated had a larger effect on the result.

A disadvantage of naïve Bayes is that, unlike machine learning methods that automatically convert whatever numeric scales you provide your inputs in, into appropriate outputs; or regression, which automatically adjusts for scale (although not for centering); with naïve Bayes, you must provide a probability for each piece of evidence. This requires validation for every possible range of output values, as described in (Chua et al. 2007).

2.3.5 Rule-based

Like best-hit, rule-based systems do not require any training. In addition to those mentioned above, which automatically construct rules, MAGPIE (Gaasterland et al. 1996) and HAMAP (Gattiker et al. 2003) use rule-based systems.

2.3.6 Bayesian

A full Bayesian analysis should give the best possible results, if priors are available. However, a full analysis is not merely computationally demanding, but usually intractable, so that some approximation is used, as in (Deng et al. 2002). PSORTb (Gardy et al. 2005) uses a Bayesian network to integrate outputs from nine SVMs.

Chapter 3: The JCVI prokaryote annotation pipeline

The J. Craig Venter Institute (JCVI) expects to process thousands of prokaryote genomes in 2012. Some genomes are processed beginning with the assembly from reads; others begin with a complete assembled genomes. The section below describes all of the steps in the pipeline after assembling a genome. (Note that eukaryotes, viruses, and metagenomic samples have different pipelines. There is also a separate prokaryotic annotation pipeline, TransAAP, for finding transporter proteins.)

JCVI's AutoAnnotate program is used by all of the Human Microbiome Project (HMP) centers:

- JCVI
- The Broad Institute
- Baylor College of Medicine
- Washington University in St. Louis, Missouri

Over the next year, these 4 centers will use AutoAnnotate to annotate at least 2000 microbial genomes.

3.1. Structural annotation: Gene calling

Glimmer 3 (http://www.cbcb.umd.edu/software/glimmer/, Delcher et al. 1999) is used to find genes using hidden Markov models (HMMs). It is specialized for prokaryotes; a different version exists for eukaryotes. Other software finds other specialized features, such as RNAs.

Genes are initially called using Glimmer 3. Later, after automatic annotation, intergenic spaces are searched for particular difficult-to-find small genes using HMMs, and start sites are refined using multiple alignments of BLAST hits.

3.2. Functional annotation

3.2.1 Homology searches (BLAST and HMM)

BLAST-Extend-Repraze (BER, http://sourceforge.net/projects/ber/) is a program that uses protein BLAST (Altschul et al. 1990) on each gene against a database of all known genes, and then does a Smith-Waterman alignment (Smith & Waterman 1981) on the top hits, extending beyond the called start and stop sites. (We blast prokaryote genes against prokaryotic, viral, and eukaryotic genes, although we haven't evaluated whether using the eukaryotic genes improves or degrades our results.) I will refer to these as BLAST hits, because that is what external sites running AutoAnnotate use instead of BER hits. BER is a post-processing step on BLAST hits to look for possible missed orthology and to correct frameshifts; this functionality is not considered in this thesis.

HMMER3 (http://hmmer.janelia.org/) is a program for building hidden Markov models (HMMs) from multiple alignments, and then matching genes against sets of those HMMs. We use two sources of HMM's: Pfams (http://pfam.sanger.ac.uk/, Finn et al. 2008) and TIGRFAMs (http://www.jcvi.org/cms/research/projects/tigrfams/overview/, Haft et al. 2003).

A JCVI program called AutoAnnotate uses the BER hits and HMM hits for each called gene to produce six outputs for each gene:

- A free-text common name, e.g., “ATP synthase F1, gamma subunit”
- A gene symbol
- An enzyme commission number (EC#) or transporter number (TC#)
- Gene ontology (GO) terms (The GO Consortium 2000)
- TIGR roles (an extension of the Riley roles, Riley & Space 1995)
- A species name or NCBI taxon number

Every gene either receives a name, or is called “conserved hypothetical protein” if it resembles other sequences believed to be genes, or simply “hypothetical protein” if it was called as a gene but does not have homology to a gene called in a previous genome. (The name "conserved hypothetical protein" is deprecated, and will no longer be assigned once homology information is stored elsewhere.) The other fields may all be left empty.

Functional annotation finds BER hits (up to 250) and any HMM hits for a gene, each of which may have different values for name, gene symbol, etc. It assigns each a rank, and takes annotation from the evidence with the lowest rank. It may pull some

additional annotation from additional hits if it is missing from the top-ranking hit (GO terms, and role ID; and EC#, if the two pieces of annotation have very similar protein names).

Protein names from some evidence sources, which are known to contain names not conformant with NCBI naming standards, are modified using from 121 to 603 regular expressions, depending on parameter settings. They are then run through the full-name rules in the Protein Naming Utility, which contains specific protein name rewrites made by JCVI's annotators on an on-going basis.

The name is then further modified based on the evidence used to assign it. Each evidence rank has an assigned naming rule. HMMs below trusted cutoff, and BLAST hits below a threshold of reliability, or from less-curated sources, either prepend "putative" to the name, or have the phrases "family protein", "domain protein", or "-like protein" appended. BLAST hits of the poorest ranks are annotated as "conserved domain protein".

3.3. Evidence ranking

3.3.1 HMM ranking

HMMs fall into one of two general types:

- Equivalogs: HMMs that span the entire length of a protein and are specific to a protein function.

- Domains: HMMs that span only a portion of a protein, to recognize a particular domain within a protein. The presence of a domain indicates membership in a family or superfamily of proteins. An *equivalog domain* spans part of a protein, but indicates a specific protein function.
- Repeats: HMMs that are like domains, but must hit multiple times to count.

We categorize Pfams and TIGRFAMs into more-specific types, called *isotypes*, that indicate the origin of the HMM (Pfam or TIGRFAM), its length (full-length or domain), and its specificity.

Every HMM has three cutoff scores: *noise*, *trusted*, and *trusted2*. Each match (or "hit") by an HMM to a gene generates a domain score. The total score for that HMM and that gene is the sum of all the domain scores from all the different places that HMM matched that gene. This is important for HMMs that identify repeats, which may need to occur some particular number of times in a gene for the protein to form a particular structure. An HMM hit also reports a match length, which is meant to indicate what fraction of the HMM matched the protein.[1]

Only HMMs with total score > *noise* are considered. They are given a starting quality score of zero, which is penalized by one for each of the following that are true:

- Match length < 75%
- Match length < 60%
- Total score < *trusted*

[1] Match length was used in HMMER2 but is always 100% in HMMER3.

- No domain score > *trusted2*

HMMs that have a quality score of 0-2 are retained, and ranked according to their isotype and quality. (Pfam isotypes of domain, family, and motif are assigned by the Pfam curators; other Pfam isotypes were assigned at JCVI and are not in the Pfam releases.)

Table 1. Rank table for HMM hits, by isotype and match quality. Note a lower "quality" score is better.

Isotype	Q=0	Q=1	Q=2
exception	1	15	23.1
equivalog	1.5	15.5	23.5
equivalog_domain	2	16	24.1
PFAM_equivalog	3	17	25.1
PFAM_equivalog_domain	4	18	26
hypothetical_equivalog	5	19	27
paralog	5.5	19.5	27.5
hypothetical_equivalog_domain	6	20.1	28
PFAM_hypothetical_equivalog	7	21	29
PFAM_paralog	7.5	21.5	29.5
PFAM_hypothetical_equivalog_domain	8	22.1	30
subfamily	40	42.1	74
PFAM_family, PFAM_subfamily	41	43.1	75
domain	47	49.1	76
PFAM_domain	48	50.1	77
subfamily_domain	51	53	78
PFAM_subfamily_domain	52.1	54.1	79
superfamily	64	66	80
repeat	68	70	82
signature	68.5	70.5	82.5
PFAM_repeat	69	71	83
PFAM_signature	69.5	71.5	83.5
paralog_domain	72.5	73.5	84.5
paralog_repeat	72.7	73.7	84.7

3.3.2 BLAST ranking

The accession to every BLAST hit is mapped to a "root accession" that is identical for every identical protein sequence, using our in-house Panda database which I created by starting with the NCBI non-redundant (NR) protein database and adding accessions for Trembl and genes we have annotated at TIGR and JCVI.. All evidence databases were previously also mapped to root accessions using Panda, so BLAST hits can be quickly matched to evidence that was annotated to the same sequence using a different accession.

Annotation is taken from the JCVI characterized protein database (Char), SwissProt, the NCBI clusters, and the set of 100,000 manual annotations made at TIGR or JCVI that I describe in Chapter 6: ("manatee"). Char is split into char_curated, char_uniprot, and char_trusted, where char_curated are annotations entered at JCVI based on publications of experimental validation of function, char_uniprot has entries imported from SwissProt that have experimental evidence for the functions annotated, and char_trusted consists of annotations imported from a variety of other sources. (FigFams are not ordinarily used.) Hits to the NCBI NR database that have no annotation in these databases are annotated as conserved hypotheticals.

NCBI clusters (PRK)[2] and FigFams (called NMPDR, following Uniprot conventions) additionally have a membership function: If a gene passes a test based on having BLAST hits to most members of the family ranked above BLAST hits to

[2] PRK clusters include only prokaryote proteins. AutoAnnotate can also use the CHL (chloroplast) and MTH (mitochondria) NCBI clusters, but we use it almost only for bacterial genomes.

members of similar families, all the hits to that family are replaced by a single hit indicating that the query gene itself is a member of the family. (This provides the same annotation, but has a superior rank.) The official membership function for FigFams implemented by their web-server is more complicated, as it involves checking for BLAST hits to members of other FigFams specific to the FigFam being tested. At the time I wrote the software, there was no documentation of the FigFam membership function other than the source code, which appears to be buggy, as one would expect it to exclude a query protein from membership if it also has BLAST hits to members of similar FigFams, whereas the code as written declares it is a member if it does have such hits. I therefore implemented my own membership function, which promotes a BLAST hit of type NMPDR to NMPDR_member if the query protein's BLAST hits include at least 15 and at least 1/3 of the FigFam's recognized members[3].

The NCBI cluster membership test requires that all BLAST hits to members of the PRK cluster outrank all BLAST hits to any other PRK clusters (Klimke et al. 2009). I tested 30 members of different PRK clusters that were annotated with an equivalent protein name by both the PRK cluster and by SwissProt. 6 of them (20%) failed this strict test due to BLAST hits to other PRK clusters. This likely gave results that conflicted with NCBI's construction of the clusters due to their unspecified BLAST score

[3] Enumerating the members of FigFams was challenging, as they are listed in a text file that specifies proteins using a wide variety of accessions, with no documentation as to their meaning, and many that are specified only by MD5 checksum values that were all calculated incorrectly. These latter could not be identified.

modification. I thus changed this membership function to require that 80% of hits to a cluster outrank all hits to other clusters.

All sequence annotation databases are searched for entries matching all BLAST hits. BLAST hits are then assigned a rank based on their identity over the match region, the fraction of the query or of the target involved in the match (whichever is smaller), and the annotation source, according to Table 2.[4]

[4] Note that NCBI BLAST's percent identity calculation counts filtered low-complexity regions as non-matching rather than excluding them from the percent identity computation.

Table 2. BLAST hit ranking table. Pink row and column intersect in the red cell, giving the rank of a BLAST hit with 50-80% identity and involving at least 65% of the length of both query and hit sequence, to a target gene that has a PRK-Validated annotation, when the list of BLAST hits passes the membership test for the PRK-Validated cluster that target gene is in. All values were chosen manually. The decimals result from re-ordering entries after the initial ranking.

% identity	**80-100**			**50-80**			**35-50**		
% length	**>80**	**>65**	**>35**	**>80**	**>65**	**>35**	**>80**	**>65**	**>35**
Source of annotation									
SwissProt	85	87.1	108	86.1	88	109	99	100	110
PRK-Reviewed_member	10	32.1	56	13	38	59	35	45	62
PRK-Validated_member	11	33	57	14	39	60	36	46	63
manatee	89	91.3	111	90	92	112	93	101	113
char_curated	9	31	55	12	37	58	34	44	61
char_trusted	54.2	54.4	55.3	54.33	54.7	58.8	54.8	83.45	61.3
char_uniprot	32.3	33.3	55.4	37.3	39.2	58.4	42.3	46.1	63.1
PRK-Provision_member	94	96	142	95	97	143	98	140	144
PRK-Reviewed	62	63	85	67	69	88	72	76	93
PRK-Validated	63	64	86	68	70	89	73	77	94
PRK-Provisional	94	96	142	95	97	143	98	140	144
NMPDR_member	10.3	32.4	56.4	13.1	38.4	59.4	35.3	45.4	62.4
NMPDR	63	64	86	68	70	89	73	77	94

If multiple hits have the same rank, ties are broken using the following rules:

- A hit with any name other than “conserved hypothetical protein” outranks a hit resulting in that name.
- A hit with more annotation (EC# and gene symbol) is favored over a hit with less annotation.
- A hit with a higher praze score (closeness of Smith-Waterman match) is favored over a hit with a lower praze score.

Chapter 4: Simplifying the ranking system

AutoAnnotate's ranking system has 13 categories of evidence source, 3 categories for %ID, and 3 categories for %length. It has a separate rank for each possible combination of these three factors, for a total of 117 ranks of BLAST hits.

Since the precision of the items near the middle of the list is believed to be around 85%, this means that the difference in precision between two adjacent ranks is about .17%. For two sources, one with a 90% accuracy and one with a 90.17% accuracy, in order to correctly identify the higher-accuracy source 90% of the time with a Student's t-test in which the statistics being compared are the number correct for each source, we expect to find a difference of .0017n in the number correct. For a two-tailed test we need a Z-value of 1.64, so we require .0017n / sqrt(2*variance/n) > 1.64. The variance of the binomial distribution being used is (.90)(.9017)n = .8115n. So we find that we require n > 1.64 * sqrt(2*.8115) / .0017 = 1229 comparisons of two sources to discern their rank correctly 90% of the time.

We can also compute the number of bits needed to construct the entire ranking table: It orders 117 BLAST hit types, and sorting 117 objects requires $\log_2(117!)$ bits. Using Stirling's approximation, ln(n!) ~ nln(n) – n, we find this is about 440 bits.

During a run, we are lucky if each gene has on average good BLAST hits to entries in two different annotation sources *A* and *B*. So a typical gene in a training set will give us one comparison. Suppose *A* is 98% reliable, and *B* is 95% reliable. This table shows the distribution of *A* and *B* being right and wrong:

Table 3. Frequency of outcomes with an annotation source *A* that is correct 98% of the time, and a source *B* that is correct 95% of the time.

	A right	**A wrong**
B right	.931	.019
B wrong	.049	.001

They will give different answers (.019 + .049) = .068 of the time. In the cases where they give different answers, *A* will be right and *B* will be wrong .019 / (.019 + .049) = .721 of the time. This means our useful outcomes have a binomial distribution with p = .721. To produce the 1 bit of information that *A* is more reliable than *B*, with 90% confidence, we need enough samples that a binomial test will give us the right answer 90% of the time. That means we need *u* useful samples (samples in which *A* and *B* disagree) where the cumulative distribution of the binomial up to *u/2* is less than .10, and the cumulative distribution from *u/2* to *u* is greater than .90. For p = .721, this means *u* = 11. To get 11 useful observations, we require .068n ≥ 11, n ≥ 162 gene comparisons.

To get the total of 440 bits needed to construct our entire BLAST ranking system, we therefore need 440 * 162 = 71,280 genes in our validation set.

We typically use about 1000 genes to construct the entire ranking table; yet this is not even enough comparisons to reliably rank two evidence sources correctly. It would take humans too much time to make enough observations to properly rank 117 evidence sources; and they wouldn't be able to remember the results anyway if they operate (as we do) by reviewing sources and gathering an impression in their heads of which is more correct, rather than counting.

As a result of the large number of rankings to be made, the rankings that come out of the meetings among the annotators are never complete (there are always cells left blank that I must fill in), and always have parts that look suspicious. In Table 2, a char_trusted 35% ID 35% length hit outranks a char_trusted 35% ID 65% length hit; char_trusted 50% ID 35% length is about the same rank as char_trusted 80% ID 80% length; and SwissProt, generally acknowledged to be the most-reliable source of manually-curated annotations, is ranked lower than most sources. For some annotation sources, 35% ID 80% length outranks 50% ID 65% length; for some, it is vice-versa. The quality>0 rankings for all of the HMMs are too high when compared to the BLAST hit table, as a below-trusted HMM hit is not trusted, while specific function is preserved 94% of the time for proteins of greater than 50% identity (Sangar et al. 2007).

I factored the rankings to reduce the number of observations needed. Two different factors go into a ranking. One is the accuracy *pga* of an annotation source ("probability good annotation", the probability that it gives a correct, acceptable name for a protein it lists). The other is the accuracy of transfer of annotation from an evidence

category, *psf* ("probability same function", the probability that the BLAST hit *h* has the same function as the query protein *q*).

Consider a query sequence *q*, a BLAST hit *h* with %ID=*I*, %length=*L*, and an annotation for that sequence from evidence source *S*. Ignore the probability that *h* has a different function from *q*, and *h* has a bad name for that function which luckily is the right name for *q*. *pga* depends on the evidence source S; *psf* depends on *I* and *L*. Combine *I* and *L* into a single %ID/%length bin *B*. We can write the probability of providing a correct annotation for *q* from *h*, *pc*, as

$$pc(S, B) = psf(B) \times pga(S)$$

We only need to estimate *psf* for each (%ID,%length) bin *B*, and *pga* for each evidence source *S*. This gives us only one set of 13 sources and one set of 3*3 hit qualities to order, instead of 117. Getting them in the right order requires only about 31 bits of information. Furthermore, we will not have problems with sparse data for any of these ranks, as we do with the larger rank table.

It can result in less precision only when there is an interaction between *pga* and *psf*. This would mean, for instance, that some evidence sources preferentially include proteins that are more or less likely to have the same function as a protein matched to it at some %ID and %length; or that some evidence sources are annotated more accurately in certain phylogenetic clades, which are likely to be a particular distance away from the query protein in %ID and %length. The most likely source of such an interaction is biased phylogenetic coverage of different evidence sources with respect to the phylogeny

of the genomes we sequence. Even if such an interaction exists, it should therefore be factored out rather than included in the model, because we don't want to train AutoAnnotate to work best for any particular type of bacteria.

This new factored ranking system will look like Table 4. For a hit to a target gene in SwissProt at 47% identity and covering 72% of the length of both genes, this table would assign a probability of 1 x .95 = .95 that the annotation was correct.

As mentioned in 0, two types of annotation sources, PRK and FigFams, have a membership criterion, satisfaction of which promotes a BLAST hit from PRK-*X* to PRK-*X*_member or from NMPDR to NMPDR_member. This membership criterion is based on sequence similarity, and so affects the probability of having the same function as the query protein (*psf*), not *pga*. Numerically, however, it will make no difference if we list the member and non-member varieties as different sources, and factor the effect of the membership function into *pga* rather than *psf*. The resulting program is considerably simpler.

Table 4. Probability that the annotation on a BLAST hit provides the correct annotation for the query sequence, as a product of the probability that it has the same function and the probability that the annotation is correct.

BLAST hit goodness:	%ID:	80-100			50-80			35-50		
	%len:	>80	>65	>35	>80	>65	>35	>80	>65	>35
	psf:	1	.99	.98	.99	.98	.96	.98	.95	.88
Source of annotation	**pga**									
SwissProt	1	1	.99	.98	.99	.98	.96	.98	.95	.88
PRK-Reviewed_member	1	1	.99	.98	.99	.98	.96	.98	.95	.88
PRK-Validated_member	1	1	.99	.98	.99	.98	.96	.98	.95	.88
PRK-Provision_member	.93	.93	.92	.91	.92	.91	.89	.91	.88	.82
manatee	.91	.91	.90	.89	.90	.89	.87	.89	.86	.80
char_curated	.97	.97	.96	.95	.96	.95	.93	.95	.92	.85
char_uniprot	.89	.89	.88	.87	.88	.87	.85	.87	.85	.78
char_trusted	.96	.96	.95	.94	.95	.94	.92	.94	.92	.84
NMPDR_member	.86	.86	.85	.84	.85	.84	.83	.84	.82	.76
PRK-Reviewed	.80	.8	.79	.78	.79	.78	.77	.78	.76	.70
PRK-Validated	.78	.78	.77	.76	.77	.76	.75	.76	.74	.69
PRK-Provisional	.64	.64	.63	.63	.63	.63	.61	.63	.61	.56
NMPDR	.36	.36	.35	.35	.36	.35	.35	.35	.34	.32

Chapter 5: Automatically constructing the ranking system

5.1. Measuring error

When we run the validation set, we get an estimate of pc(s,b) for every (s,b) combination, that says how often an annotation from source *s* to a hit with %ID=*I* and %length=*L* was judged to be correct by the validation program:

Equation 1. Estimate of fraction of corrections that are correct for a (source, bin) combination.

$$\text{right}(s,b)/\text{trials}(s,b) = \text{pc}(s,b) \approx \text{psf}(b) \times \text{pga}(s)$$

where trials(s,b) is the number of hits from source *s* and bin *b* that were evaluated, and right(s,b) is the number of them judged to have the right annotation. The "≈" means we want to set psf(b) and pga(s) so that it is an equality, but may have to settle for getting close.

Call the set of all bins B. Call the set of different sources S. I'll also use B to mean the number of bins in B, and S to mean the number of sources in S. (This should not be a problem for Perl programmers. S is a list when evaluated in list context, and a scalar when evaluated in scalar context.)

We want to find psf(B) and pga(S) that minimize some error measure. The error of predictions in one (s,b) pair is the number of hits in that bin that were right, minus the number predicted to be right using particular values for psf(b) and pga(s):

Equation 2. Error of prediction based on new ranking method.

$$Err(s,b) = right(s,b) – trials(s,b)*psf(b)*pga(s)$$

(I used right(s,b) – trials(s,b)*psf(b)*pga(s) instead of pc(s,b) - psf(b)*pga(s) because the former gives more importance to settings the values for bins and sources that have a lot of hits.)

This is the sum of squared error over (s,b) pairs, which should be minimized:

Equation 3. Sum of squared error, to be minimized to set *psf* and *pga* vectors.

$$Err2 = \sum_{s \in S} \sum_{b \in B} \left(\text{right}(s,b) - \text{trials}(s,b) \times \text{psf}(b) \times \text{pga}(s)\right)^2$$

5.2. Minimizing error

Where the error is minimal, all the partial derivatives are zero:

Equation 4. Partial derivatives to be set to zero to minimize Equation 3.

$$\frac{\partial Err2}{\partial psf(b)} = 0 = 2\sum_{s \in S} \left(\text{right}(s,b) - \text{trials}(s,b) \times \text{psf}(b) \times \text{pga}(s)\right)\left(-\text{trials}(s,b) \times \text{pga}(s)\right)$$

$$\frac{\partial Err2}{\partial pga(s)} = 0 = 2\sum_{b \in B} \left(\text{right}(s,b) - \text{trials}(s,b) \times \text{psf}(b) \times \text{pga}(s)\right)\left(-\text{trials}(s,b) \times \text{psf}(b)\right)$$

So we have S+B unknowns and S+B equations. This means we can solve for the psf(B) and pga(S) that will give us the minimum squared error. Not only will our ranking be more accurate, but we'll have a calibrated probability-of-correctness of each annotation. The only catch is that it requires automatic validation to work well.

Because we have fewer parameters to determine than with the old ranking system, we could break up %ID, %length into more bins. Instead, I wrote a program to interpolate between bins. The program could use some sort of regression over %ID and %length to determine their contributions to *psf* as a continuous function. But linear regression would be inappropriate, as this contribution is nonlinear; and linear piecewise

regression may be appropriate, but probably no more accurate than linear interpolation between these bins.

To solve this difficult system of equations, I used Newton's method. We know $f(x+dx) - f(x) \sim f'(x)dx$, and wish to find x such that $f(x+dx) = 0 = f(x) + f'(x)dx$. This is given by $dx \sim -f(x) / f'(x)$. Iterating this for all s+b equations above until convergence provides optimized values for all of the variables. In practice, if the vectors *psf* and *pga* are initialized with either .95 or with priors based on the training set, the program always converges within 400 iterations, which takes less than a second.

Chapter 6: Building a gold-standard annotation set

We need a set of genes with known correct annotations, both to train and to test an annotation system. There are several available sets:

- Genome annotations for model organisms, such as E. coli
- SwissProt
- The NCBI ProtClust clusters
- The NMPDR FigFams

There are two problems with these sources. First, none of them give protein names according to JCVI naming conventions; so results would not be very convincing to JCVI annotators. Second, I was at the same time using all of these as the annotation sources that I wanted to rate. Any source would test as very accurate when it is its own gold standard. Third, some of them, especially the first two, were used as the basis for constructing TIGRFAMs and for constructing the other annotation sources, so the measured accuracy of FigFams as measured by comparison Swissprot might depend more on the degree to which NMPDR used SwissProt annotations, than on the accuracy of the FigFams.

I gathered 100,000 manual gene annotations made at JCVI and TIGR over the past 5 years. I began with 686 manually-curated bacterial genomes. About 200 of them have been removed from our databases and are no longer accessible. Several of them had project names, NCBI taxon IDs, or annotation indicating they were not really bacterial genomes. A few had almost every gene manually curated; most had only a dozen or so manually curated.

I wrote a Perl program to scan all of the annotations for these genomes, and find those that were made by humans rather than by a computer program, and which had more than one GO annotation (Gene Ontology Consortium 2000). (Filtering on GO annotations is intended to select for annotations made more carefully.) Out of roughly 100,000 manually-annotated genes, this selected 63,777.

To reduce redundancy, I constructed a BLAST database from these 63,777 genes, blasted them all against each other, and removed any gene that was more than 85% identical to another gene with an alphabetically-earlier name. This removed 14,260 genes, leaving 49,517.

Initially, I assembled these genes into two genomes, one gram-positive, and one gram-negative. AutoAnnotate uses SignalP 3.0 (Bendtsen et al. 2004), a program to identify signal peptides; this program must know whether a gene comes from a gram+ or gram- bacteria. However, this was unworkable, because AutoAnnotate runs on a single processor, and a 30,000-gene genome takes days to run.

I then instead assembled these genes into 55 virtual genomes, organized taxonomically, so as to put genes from the same species or genus in a single genome. (This step was extremely time-consuming, and so far has no impact on the result, since each gene is evaluated separately without regard to properties of the organism it resides in other than whether it is gram-positive or gram-negative.) Then I wrote a script to run AutoAnnotate on these 55 virtual genomes among different nodes in our computational grid, so they could be done in a somewhat parallel manner.

Chapter 7: Comparing protein names

AutoAnnotate produces protein names, EC numbers, GO terms, and TIGR role IDs. Of these, JCVI believes users care about the protein names, and a little bit about the EC numbers. Most annotations don't have EC numbers. Therefore, getting the protein names right is the main criterion on which AutoAnnotate's performance is judged.

When using the gold-standard annotation set to assess the accuracy of AutoAnnotate, or to assess the accuracy of a particular evidence type (and provide a rank for it), it is necessary to compare the protein names assigned by AutoAnnotate, to those given by annotators. One annotator might describe a protein as “3-dehydroquinate dehydratase, type I”, while the annotation taken from one of the evidence databases might describe it as “Dehydroquinase class I”, or “Type I 3-dehydroquinase”. These names will be judged different, and the annotation as inaccurate, unless we can provide an automated method to tell that they refer to the same protein function.

7.1. Simplifying protein names

I began by writing code to strip uninformative words and phrases from protein names, such as “protein”, “putative”, “-like”, or “homolog”. In many cases, it’s a judgement call whether to keep a phrase that makes a distinction, such as “homolog”,

“family”, or “similar to”. Is it better, or worse, to group annotations identifying a protein as being similar to protein Y together with annotations identifying it as protein Y? If you have candidate annotations that differ greatly in function, it's better to group similar annotations. But if most of the candidate annotations are already Y-like, then the task is just to discriminate between Y and Y-like; and grouping those together makes that impossible.

Many changes are not controversial: mapping synonyms into one standard word, including Roman numerals to Arabic, and English spellings to American; removing comments appended in parenthesis or giving identifying information such as gene symbol and EC number; and converting letters to lowercase.

7.2. Correcting misspellings

I approached spellchecking in several ways.

I made a list of common misspellings in annotations, and code to correct those.

I used the spelling corrections built in to the JCVI Protein Naming Utility (PNU) (Goll et al. 2009).

I investigated using GNU Aspell, an automatic spellchecker. I did not use it because it requires a dictionary of all allowable words. I had no such dictionary; nor would compiling one from i.e. Uniprot work, since people continue to make new annotations that may include new gene symbols, locus tags, taxon identifiers, or strange abbreviations.

I made a list of commonly-misspelled words, and wrote a program that respells a query word as a correctly-spelled target word if it:

- is from one less to one more letter in length
- has no letters not in the target
- is missing at most 1 of the letters in the target
- starts and ends with the same letters as the target
- is missing at most 3 bigrams (letter pairs) of those in the target

Removing a letter removes 2 bigrams and adds 1; inserting a letter removes 1 and adds 2; swapping 2 adjacent letters changes 3 bigrams. That is why I allowed the query (potential misspelling) to be missing up to 3 bigrams present in the target (correct) word. Just using the target words 'conserved', 'hypothetical', 'protein', 'phosphatase', 'putative', and 'transporter' corrected 176 distinct misspellings in BioThesaurus 6. Analysis of the word 'protein' showed that roughly ¼ of all possible misspellings under these rules were found. Here are the misspellings of the word “protein” that were corrected:

Table 5. Misspellings of 'protein' meeting constraints found in existing annotations.

peotein	proein	proteinn	protrein
perotein	proetein	proteion	prottein
pnrotein	proetin	proten	prptein
pootein	proiein	protenin	prrotein
porotein	proitein	proteon	prtein
portein	prootein	proterin	prtoein
potein	propein	proterin	prtotein
pprotein	proptein	protetin	ptotein
preotein	prortein	protiein	ptrotein
pretein	proteein	protien	
pritein	proteiin	protin	

The Broad Institute distributes a Python tool called PIDGIN (http://genepidgin.sourceforge.net/) for judging when two protein names mean the same thing. It was not available until after I finished the above work, however. Study of the technique used by "Pidgin compare" does not convince me it will perform any better.

, and an ontology saying when one protein name is a child of another (describes a subset of the set its parent refers to). I describe them below.

7.3. Protein name thesaurus

Validation relies on being able to group annotations into sets of annotations with the same meaning. Because protein names are non-standards, many annotation sources don't provide EC numbers or gene symbols or GO terms, and because gene symbols are

non-standard, this is difficult. Also, while GO terms are standard, their assignment to a particular protein is not; therefore, GO terms can't be used to determine whether two annotations refer to the same protein even if they're present.

I therefore built a thesaurus of protein names judged to probably be equivalent. I took assertions that names were equivalent from the following sources. Protein names from all of them were used only if they passed a filter to screen out useless names ('hypothetical', 'protein fragment') non-descriptive names ('DUF2349'), or nonsensical names (‘brl34_jg’, of which there are many in BioThesaurus). The thesaurus records only names, not any protein families that the names may be taken from.

7.3.1 Enzyme commission synonyms

The Enzyme Commission provides recommended names and alternate names for enzymes that they have given an EC number to. These names can be downloaded from the ENZYME database at ftp://au.expasy.org/databases/enzyme/. This provided less than 8000 synonym sets.

7.3.2 The JCVI Protein Naming Utility

When annotators find a protein name that's misspelled or misleading, and change it to another name indicating the same function, they record that change in a database called the PNU (Protein Naming Utility). This provided 11,511 synonym sets.

7.3.3 COGs

I tried to construct synonyms from COGs (Tatusov et al. 2000), by declaring that the annotations in SwissProt of proteins from the same COG family were synonyms. Inspecting the results showed this method often grouped together proteins with different functions in SwissProt, so I stopped using it.

7.3.4 TIGRFAMs (Haft et al. 2003)

Each TIGRFAM gives both a family protein name, and an extended name (meaning a more-descriptive name that some annotators object to on aesthetic grounds). These were therefore declared to be synonyms. This provided 592 sets of synonyms.

I also tried looking up annotations on all of the proteins used as seeds in constructing a TIGRFAM, and declaring them synonymous. Inspecting the results showed this to be unreliable, so I stopped using it.

7.3.5 The InterPro protein hierarchy

When looking up protein family information on InterPro's website, you can sometimes find information on its InterPro parent and child families. This information is taken from a 6,211-line file that can be found at ftp://ftp.ebi.ac.uk/pub/databases/interpro/ParentChildTreeFile.txt. It lists an InterPro family name, equivalent family names in other protein family systems, and the InterPro families that are subsets of the first InterPro family. This provided 2747 synonym sets.

7.3.6 Uniprot

Uniprot often lists synonyms for names, tagged with "AltName". I parsed Uniprot and added all synonyms there. This provided 21,803 sets of synonyms.

7.3.7 BioThesaurus (Lui et al. 2006)

BioThesaurus gives millions different protein names, and puts them in groups that are believed to mean the same thing. Most of these millions of names have no useful synonyms other than the same name re-written in uppercase or lowercase, so that statistic is misleading. From 30 million protein names, I was able to extract 177,000 sets of usable synonyms that passed my filter. Many of these were not very useful, such as "DEHA0C04686g = deha2c04092g".

7.4. Protein name ontology

Having sets of equivalent protein names does not address the situation where one annotation gives a name that specifies a subset of the proteins described by another name. For instance, according to Interpro, "Septin 7" is a subclass of "Septin", which is a subset of "Cell division/GTP binding protein". If one annotation assigns the name "Septin", and another assigns the name Cell division/GTP binding protein", they are in partial agreement, and should not be scored the same as if they were completely different.

Therefore, I built a hierarchy of protein families, giving subset and equality relations between different protein families. Below are the sources I used or tried to use.

7.4.1 PRO, the Georgetown protein ontology (Natale et al. 2007)

PRO does not provide information about what protein families are subsets of other protein families. It provides information about what proteins are variations of other proteins. It mostly contains information on alternate splicing of eukaryotic genes. Therefore, I decided it was not worth my time to parse information from this database.

7.4.2 PIRSF (Wu et al. 2004)

PIRSF provides subfamily/superfamily relationships for some of its families. This provided 1153 subset relationships.

7.4.3 The InterPro protein hierarchy

Using the same file ParentChildTreeFile.txt as for the thesaurus, I instead extracted equivalence and subset relations for the named protein families. This provided 4534 subset and 5035 equivalance relations.

7.4.4 Protein familes

I downloaded the data describing the protein families COG (Tatusov et al. 2000), HAMAP (Lima et al. 2008), InterPro (Hunter et al. 2009), KEGG (Kanehisa & Goto 2000), NMPDR (FigFam, Overbeek et al. 2005), PIRSF (Wu et al. 2004), and NCBI ProtClustDB clusters of type PRK (Prokaryote; Klimke et al. 2009). This provided 118,390 protein families.

I then compared each protein family to all other protein families that it shares at least one protein with, and declared it a subset of another family if more than some threshold percentage of its members were in that family. PIRSF uses a threshold of 75% for that purpose.

7.4.5 Uniprot

Uniprot provides family-membership information telling which proteins are in HAMAP, HSSP, InterPro, KEGG, NMPDR, PANTHER, PIRSF, PROSITE, Pfam, STRING (von Mering et al. 2007), and TIGRFAM (Haft et al. 2003) families. I parsed the swissprot data file, and created a database listing all those protein family memberships. I intend to construct families from this data and incorporate them into the ontology as was done for the protein families listed in the previous section. So far, I have done this for Pfams and TIGRFAMs, and use Uniprot to supplement the original-source data files for KEGG, HAMAP, NMPDR, and PIRSF.

Using a threshold of 80% and at least 3 members per family, the original source data for protein families plus Uniprot provided 307,960 subset relations. Upping the threshold to 90% still provided 252,712 subset relations.

7.4.6 Construction of equivalence classes

In addition to the equivalence classes specified by InterPro, I declared that two families were equivalent if they were each subsets of each other. I constructed a list of all equivalence families thus created. This was something the annotators wanted anyway, so

they could compare the protein names on equivalent families, and decide which data sources gave the best and most-accurate protein names.

Note that my definition of subset is not the same as in set theory. It is not even transitive: If A < B (A is a subset of B), that means, say, 90% of the proteins in A are in B. Likewise for B < C. But this guarantees at most that 81% of the proteins in A are in C. Similarly, the equivalence relation '=' induced by this definition of subset is not transitive.

I found that constructing equivalence relations between families this way sometimes gave bad equivalence relations. It can happen that one HMM identifies a domain or subunit, and another family identifies the complete protein, and there are almost no proteins with the domain that are not in the family. This results, for instance, in the following names being declared equivalent:

Table 6. An example of HMMs intended to be less functionally specific that nonetheless hit exactly the same proteins in SwissProt as the more-specific TIGRFAM.

Source	Accession	Protein name
Pfam	PF01624	MutS domain I
Pfam	PF05188	MutS domain II
Pfam	PF05192	MutS domain III
Pfam	PF05190	MutS family domain IV
Pfam	PF00488	MutS domain V
TIGRFAM	TIGR01070	DNA mismatch repair protein MutS

These Pfam domains will be scored, as a source, as being as precise as the function-specific protein family TIGR01070. This results in the precision of Pfam domain names being over-estimated. However, it does not appear to be a problem when considering only BLAST hits.

Chapter 8: Training and testing on the validation set

8.1. Training and testing sets

The validation set was divided into two parts, a large part for training and a small part for testing. Each gene in the validation set had a name ending with a 5-digit number. All genes with names not ending in '1' were used for training; all genes with names ending in '1' were used for testing.

8.2. Grouping BLAST hits by annotation

The simplest approach would be to train by rating each BLAST hit individually as being correct (the same as the name assigned by the manual curator) or incorrect. Doing so still resulted in most BLAST hits being judged incorrect, whether they were or not, as the number of minor variations on protein names is unlimited.

Therefore, when training, I grouped all BLAST hits according to their protein names and other annotation. Hits with protein names that were judged to be synonymous after stripping and running through the thesaurus, or with identical gene symbols or EC numbers, were placed in the same group, provided that gene symbols and EC numbers did not conflict.

Hits that could not be assigned to a group based on perfect match, had a similarity metric computed between their protein name, and all the protein names of all grouped BLAST hits. This metric counted the fraction of trigrams found in either name that occurred in both, plus the fraction of GO terms associated with either hit that were associated with both, and so produced a score from 0 to 2. If the hit had a similarity > 0.8 to any grouped hit, it was placed in the group of the hit that its similarity to was greatest.

Hits that still could not be grouped were ignored for training.

8.3. Grouping BLAST hits by sequence

I also wrote methods for grouping BLAST hits by sequence similarity. One such method grouped together all BLAST hits that were themselves matched by equivalog HMMs. Another used k-means clustering to group BLAST hits based on trigam counts in their protein sequence, followed by generation of a multiple alignment using MUSCLE, and then a position-specific scoring matrix (PSSM) built from the multiple alignment using PSI-BLAST. Each gene was then re-grouped into the group whose PSSM it best matched. These methods performed very well at grouping proteins of the same function together, regardless of their annotation. However, they were very slow (taking roughly 100 times as long to run). More importantly, the method proposed here requires grouping BLAST hits together based on their annotation, and whether or not they truly have the correct annotation, or the same function as each other, is estimated statistically for their annotation type, not individually for each hit. Grouping by sequence can be useful for annotation – possibly more useful than what was done here – but it does

not fit easily into this framework, and certainly cannot be done at the same time as grouping by annotation.

8.4. Estimating *psf* and *pga* for individual BLAST hits

I retained some of this functionality, by noting when BLAST hits were annotated in Uniprot as having HMM hits to them, where the same HMM hit the query sequence and thus was another piece of evidence, and was also grouped based on its name. I marked such BLAST hits as having correct annotation if they were in the same group as the HMM that hit them, and incorrect annotation otherwise. I marked them as having the same function as the query if they were hit by an equivalog HMM that also hit the query, and having a different function if they were not hit by such an equivalog.

I also estimated *pga* for BLAST hits by finding the ontology group associated with their name, finding any protein groups that Swissprot said that BLAST hit was a member of using sequence-based methods, finding the number of proteins in SwissProt that are members of both the ontology group and the protein group, and then taking the maximum of that number divided by the size of the name ontology group or the protein group.

8.5. Judging the correctness of annotation groups

After grouping, each group is checked to see if its names specify a group in the ontology. For each group that can be matched to an ontology group, its parent ontology groups are recorded.

To choose which group was correct, AutoAnnotate groups the query protein in the same manner as other BLAST hits. The group that it is placed in is judged to be correct, with a probability of correct judgement (PCJ) based on the similarity score of the query protein to the group. If no group was judged to be correct, that gene was skipped for training. All HMM equivalogs that were placed in the correct group (according to their name) are recorded, and called *correct equivalogs*.

AutoAnnotate then goes through all of the groups again, to look for groups labeled as wrong that could still be correct for a number of reasons. First it assigns each hit in each group a probability of being correct that is the fraction of the correct equivalogs that hit that BLAST hit, and a probability based on the top similarity of its name to names in the correct group. These two probabilities are combined according to Equation 5.[5]

[5] Oddly, I found no guidance in the mathematical literature on how to combine multiple independent probability estimates for the same event – probably because this is regarded as theoretically nonsensical, despite its empirical necessity. I had to come up with Equation 1Equation 5 myself.

Equation 5. Approximating a new probability based on multiple independent probability judgements of the same event.

$$\frac{\prod_i p_i}{\prod_i p_i + \prod_i (1 - p_i)}$$

Each group is then given a probability, computed using Equation 5 again over the probabilities just assigned to all of its hits. For all groups other than the chosen group, this probability is multiplied by *PCJ*, since it is conditional on the correct group having been chosen.

All groups assigned a probability less than .5 are removed from the training set if they contain a name that is a substring of a name in the correct group, or if the group is a child of the correct group according to the ontology, because either condition indicates the group is likely to contain annotation that is nearly correct. Any group that is a parent of the correct group is marked as correct, with a probability equal to PCJ.

Training is done on all groups that at this point have a probability less than .4 or greater than .6. Each value *right(s,b)* in Equation 1 is computed as the sum, over all BLAST hits in bin *(s,b)*, of the probability of that hit's annotation group.

I mentioned in 8.3 that I could in some cases judge a BLAST hit to have the same function or a different function from the query protein based on how many of the same HMMs hit it; and also, whether its annotation was good or bad based on whether it was in

the same annotation group as the HMMs that hit it. In cases where the confidence in the estimate was high (*psf* or *pga* < .2 or > .8), I excluded such BLAST hits *h* from the error minimization described in Section 5.2, and applied them afterwards to adjust the *psf* for their %ID/%length bin (if *psf(h)* was known), and the *pga* for their annotation source (if *pga(h)* was known). Equation 6 shows how this was computed for *pga*. *psf* is updated analogously.

Equation 6. Re-computing *pga* using BLAST hits *h* with reliable estimates for *pga(h)* .

$$pga(source) = \frac{right(source) + \sum_{h \in BLAST} pga(h)}{trials(source) + |BLAST|}$$

Running AutoAnnotate on my training set, and then performing these computations on the gathered data, produced these estimates for the vectors *psf* and *pga*:

Table 7. *PSF*: Estimated probability that a protein BLAST hit with the given %identity and %length values has the same function as the query sequence.

		%ID		
% length	**>80**	**>50**	**>35**	**>25**
>80	.993	.997	.994	.974
>65	.810	.979	.970	.966
>35	.940	.978	.959	.913

The values are reasonable, with the exception of the >80%ID / >65% length and >80%ID / >35% cells, which had only 35 and 10 samples (versus thousands for most other cells). I changed the leftmost column (>80% ID) to .998, .966, and .940, to maintain monotonicity going left-to-right and up-to-down in the table. I changed >50% ID, >35% length to .975, to make the drop from 65% to 35% length more in keeping with the other columns.

The adjustment for *PGA*, however, adjusted all values too strongly downward. I believe that estimating values of *pga* for individual hits based on intersections of ontology groups and protein groups often produced values significantly below 1 even for correct annotations. Also, the ontology database that this calculation used contained many erroneous entries due to a bug that assigned the wrong identifiers to many protein families. I therefore re-ran the computation of *PGA* without holding out BLAST hits with estimated values of *pga*, and without doing the adjustment afterwards described by Equation 6. This gave the vector for *PGA* in Table 8.

Table 8. Estimated probability of good annotation by annotation source.

Source of annotation	pga	samples
manatee	1	20256
char_curated	1	645
SwissProt	.996	3677
PRK-Reviewed_member	1	119
PRK-Validated_member	1	95
PRK-Provisional_member	.999	327
PRK-Reviewed	.998	129
char_uniprot	.990	1401
PRK-Validated	.998	81
NMPDR_member	.965	2537
PRK-Provisional	.972	302
char_trusted	.850	1272
NMPDR	.771	1256

Note that manatee is the same database used for the validation set. However, manatee hits of 100% identity were excluded from the training, ensuring that no manatee entry was ever used to annotate itself. Its high performance can be partly attributed to the fact that the manatee annotations were made using JCVI-style protein names. For our purposes at JCVI, however, that is not something we want to compensate for in our ranking. The large number of manatee samples also suggests that the training set contained many similar proteins.

The low scores of non-member hits from NMPDR should not be seen as reflecting the annotation quality of NMPDR in general. As you will recall, that number is reduced by the probability that a NMPDR hit that does not satisfy the membership function does not have the same function. Also, the reason JCVI does not currently annotation from NMPDR annotations is not because they are inaccurate, but because their naming conventions are very different from JCVI's. This would give them a low *pga* value. Again, for JCVI's purposes, this unfair low score is desirable.

I downgraded all the non-member PRK sources to .94 or less, based on their lower scores in previous tests.

8.6. Testing

Testing is simple compared to training. The values of *psf* and *pga* computed from the training set are entered into AutoAnnotate and used to run the new ranking method on the test set. The accuracy is judged by the number of times that the annotation group that the manually-curated annotation for the query gene was assigned to is the same as the group that the top-ranked annotation is in. We then compare the results for new and old methods. I turned off HMMs for the testing run, as leaving them on would result in HMMs (which are more reliable than BLAST hits) being used on different query proteins in the two tests. I also turned off the use of the similarity metric for grouping annotations, as I observed that using name similarity sometimes resulted in cases where one method got the correct answer and the other chose a similar but incorrect annotation, but still received full credit for it because their names were similar.

"False positives" are cases where the ranking system chose an annotation group, but did not assign the curated annotation to any of the annotation groups. These are not necessarily errors, as the curated annotation might properly belong to the chosen annotation group. "False negatives" are cases where the system assigned the curated annotation to an annotation group, but was unable to choose an annotation group with high confidence. "True errors", the remaining cases, are the worst kind of error, where the curated annotation was assigned to one annotation group and the system chose another.

Table 9. Test results, given as fraction of sample.

Ranking method	Samples	Correct	False positives	False negatives	True errors
original	4508	.615	.082	.064	.239
new	4508	.724	.082	.068	.126

A one-tailed binomial test of the chance of getting $4508 \times .724 = 3264$ correct answers from 4508 samples if the chances of getting a correct answer were .615, produces a Z-score of (.724 - .615) / sqrt(.615 * (1-.615) / 4508) = 15.04, which indicates a probability of 4.4E-54. Hence, the new method produces more correct answers.

The true accuracy of AutoAnnotate is higher in both cases, as annotations that were not judged correct can still be correct due to either being different ways of describing the same function, or describing the same function at a greater or lesser level

of specificity. Also, AutoAnnotate normally uses HMMs to annotate about one-third of all proteins, with high reliability.

Chapter 9: Conclusions and Future Work

9.1. Discussion

Breaking the reliability of a transitive annotation into two independent components, and automatically assigning values to the parameter vectors *PSF* and *PGA*, both appear to work as hoped for. The values assigned are largely congruent with expectations about the reliabilities of different degrees of BLAST hit similarity and of different evidence sources.

Using the two vectors *PSF* and *PGA* is less controversial than assigning their values automatically. The new method performs better when measured by the metric that it is designed to optimize, which is not surprising. If the validation set, or the procedure for identifying correct and incorrect annotations, is biased in some way, the automatic construction of *PSF* and *PGA* could optimize for those biases. In the most extreme case, about half of the annotations in *char_trusted* are different components of taxon-antitoxin systems, which may be difficult to identify using the thesaurus.

The automatic construction of *PSF* should be less controversial, since no one has firm opinions on what the relative reliabilities are of different combinations of percent identity and length in a BLAST hit, and since bias in the validation set or in the

construction of the thesaurus is likely to affect all of the different %ID / % length categories equally. The automatic construction of *PGA* is more controversial. One issue is that SwissProt is a standard curated annotation set, and was directly or indirectly used to construct or check many of the other curated sets. Other sets could therefore be biased to have annotation concordant with SwissProt, whether it is correct or incorrect. Another issue is that the validation set consists of manually-curated genes; and owing to the increasing rarity of manual curation, most of these genes were annotated more than 4 years ago. The validation set could therefore be biased to favor older annotation sources that have not been corrected. The validation set was culled to not have proteins with more than 85% sequence identity to each other; but it could still contain many orthologous proteins from related organisms, giving *manatee* a higher score than it deserves. The values produced by the optimization process described in section 5.2 therefore cannot be taken at face value.

On the other hand, as shown in Chapter 4:, human judgements cannot be taken at face value either. It takes a great deal of effort for a human to study even a hundred cases for each evidence source; and that is a tiny fraction of even the smallest evidence source (let alone the 18 million entries in Trembl release 2011_11). There is no reason to think that the bias inherent in picking out such a small sample is less than the bias of the thesaurus at recognizing the names of particular protein families, or the bias of the validation set.

A larger problem is the tradeoff between accuracy and adherence to naming conventions. PDB may be the most accurate of all curated annotation sources, as every

protein in PDB has a known three-dimensional structure; and when a function-specific name is provided, it is highly probable that that name was assigned on the basis of functional assays performed prior to the expensive task of determining a protein's structure. Yet JCVI does not use PDB for names at all anymore, due to dislike of PDB's lack of naming conventions and proclivity for extremely long, descriptive names.

In the absence of good high-throughput methods for validating the accuracy of annotations, annotators have focused instead on naming conventions. When annotators from different centers convene, as at the NCBI Annotation Workshop held at JCVI in April 26-27, 2010, discussion of structural annotation focuses on its accuracy, because there are methods for evaluating its accuracy; while most discussion of functional annotation is about naming conventions. This is made worse by two factors: First, GenBank has hundreds of restrictions on the names that may be used in functional annotations submitted to them, and will reject and send back genomes with any violations; but they have not way to screen functional annotation for accuracy. Sequencing centers are thus motivated to prefer names that conform to GenBank standards over names that are more accurate, and to devote their time to protein naming conventions rather than to accuracy. Second, annotators have little contact with end users. This has enabled an annotation culture to develop that focuses most of its attention on naming conventions that may have no importance for end-users, and that sets standards for accuracy without reference to user requirements. It is impossible to discuss the optimal tradeoff of false positives versus false negatives when many annotators believe that no false positives are acceptable.

It is therefore difficult to apply the work in this thesis. The purpose of the work is to improve on the ability of humans to rank different annotation sources; yet ultimately those human judgements are the standard by which the work's performance will be measured. Any improvement in performance will thus be interpreted as a degradation in performance, since it means lesser agreement with human judgement. It is easy to see why all other recent work on automation of functional annotation assigns GO terms rather than protein names. GO terms are standardized, and so an evaluation can more authoritatively say whether the assigned GO terms are the same as those in the validation set or not. But we are judged, by GenBank and by other annotation centers, more on our protein names than on GO terms; and we have no validation set of GO annotations of known reliability.

One reasonable experiment would be to compare scores with automatically-generated vectors for *PSF* and *PGA,* versus automatically-generated *PSF* and annotator-produced *PGA*, and annotator-generated *PSF* and automatically-generated *PGA*. This would indicate how much of the improvement came from the choice for *PSF*, and how much came from the choice for *PGA.*

It is my expectation that considerable improvement comes from the automatic construction of *PGA*, particularly because SwissProt is a large source of what I believe is good annotation. The JCVI ranking system assigns SwissProt a ranking so poor that it is almost never used for annotation, and it provides only generic, hesitant names when used. It is possible that most of the gains of this system come simply from increasing the use of

SwissProt for annotations. If this is so, manual "correction" of the automatically-constructed *PGA* vector will destroy most of the gains made.

9.2. Future work

The test should be re-run after the thesaurus-generating code is improved to incorporate Trembl accessions, which would improve the coverage of the protein names and protein families used to construct the thesaurus. Alternately, the test could be run using GO terms, if a positive performance would be interpreted as an indication of similar performance with protein names.

Estimating the probability that a BLAST hit has the same function using only percent identity and length of hit ignores the fact that different protein families have different numbers of orthologs, and hence different criteria should be applied to each protein family to determine whether a BLAST hit has the same function. The phylogenetic distance between the query and the BLAST hit, as well as the diversity of the protein family it is in and the amino-acid Hamming distance to nearby families, should be taken into account.

The current approach still takes all annotation from a single piece of annotation. It does not make use of evidence groups other than for validation. Annotation could be assigned with higher priority if multiple, independent pieces of annotation agreed. Also, if BLAST hits were grouped by sequence rather than by annotation, a multiple alignment of the different groups could be used to re-evaluate query/hit similarity in a way that emphasized conserved sequence.

References

References

SF Altschul, W Gish, W Miller, WE Myers, DJ Lipman (1990). Basic local alignment search tool. *J Mol Biol* 215 (3): 403–410. doi:10.1006/jmbi.1990.9999. PMID 2231712. http://www-math.mit.edu/~lippert/18.417/papers/altschuletal1990.pdf.

MA Andrade, NP Brown, CS Leroy, S Hoersch, A de Daruvar, C Reich, A Franchina, J Tamames, A Valencia, C Ouzounis, C Sander (1999). Automated genome sequence analysis and annotation. *Bioinformatics* 15, 391-412.

A K Aziz, D Bartels, AA Best, M DeJongh, T Disz, RA Edwards, K Formsma, S Gerdes, EM Glass, M Kubal, F Meyer, GJ Olsen, R Olson, AL Osterman, RA Overbeek, LK McNeil, D Paarmann, T Paczian, B Parrello, GD Pusch, C Reich, R Stevens, O Vassieva, V Vonstein, Andreas Wilke, O Zagnitko (2008). The RAST Server: Rapid Annotations using Subsystems Technology. *BMC Genomics* 9:75, doi:10.1186/1471-2105-9-75.

M Biswas, JF O'Rourke, E Camon, G Fraser, A Kanapin, Y Karavidopoulou, P Kersey, E Kriventseva, V Mittard, N Mulder, I Phan, F Servant, R Apweiler (2002). Applications of InterPro in protein annotation and genome analysis. *Briefings in Bioinformatics* 3, 285–295. (RuleBase)

J Boekhorst, B Snel (2007). Identification of homologs in insignificant blast hits by exploiting extrinsic gene properties. *BMC Bioinformatics* 8:356-362.

JM Chandonia, SE Brenner (2006). The impact of structural genomics: expectations and outcomes. *Science* 311: 347–351.

HN Chua, WK Sung, L Wong (2007). An efficient strategy for extensive integration of diverse biological data for protein function prediction. *Briefings in Bioinformatics* 23(24):3364-3373, doi:10.1093/bioinformatics/btm520.

JCA Chung, G Dinkov, WC Barker (2004). PIRSF: Family classification system at the Protein Information Resource. *Nucleic Acids Res.* 32(Database): D112-114.

V Curwen, E Eyras, T Daniel Andrews (2004). The Ensembl automatic gene annotation system. *Genome Res.* 14: 942-950.

AL Delcher, D Harmon, S Kasif, O White, & SL Salzberg (1999). Improved microbial gene identification with GLIMMER. *Nucleic Acids Research* 27(23): 4636-4641.

M Deng, T Chen, F Sun (2004). An integrated probabilistic model for functional prediction of proteins. *Journal of Computational Biology* 11(2–3):463-475.

R Eisner, B Poulin B, D Szafron, P Lu, R Greiner (2005). Improving Protein Function Prediction using the Hierarchical Structure of the Gene Ontology. *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology 2005*.

O Emanuelsson, H Nielsen, S Brunak, G von Heijne (2000). Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *J.Mol. Biol.* 300:1005-1016.

O Emanuelsson, S Brunak, G von Heijne, H Nielsen (2007). Locating proteins in the cell using TargetP, SignalP, and related tools. *Nature Protocols* 2:953-971.

R.D. Finn, J. Tate, J. Mistry, P.C. Coggill, J.S. Sammut, H.R. Hotz, G. Ceric, K. Forslund, S.R. Eddy, E.L. Sonnhammer, A. Bateman (2008). The Pfam protein families database. *Nucleic Acids Research* Database Issue 36:D281-D288.

W Fleischmann, S Möller, A Gateau, R Apweiler (1999). A novel method for automatic functional annotation of proteins. *Bioinformatics* 15:228–233.

T Gaasterland, C Sensen (1996). Fully automated genome analysis that reflects user needs and preferences-a detailed introduction to the MAGPIE system architechure. *Biochimie* 78:302-310.

JL Gardy, MR Laird, F Chen, S Rey, CJ Walsh, M Ester, FSL Brinkman (2005). PSORTb v.2.0: expanded prediction of bacterial protein subcellular localization and insights gained from comparative proteome analysis. *Bioinformatics* 21:617-623.

A Gattiker, K Michoud, C Rivoire, AH Auchincloss, E Coudert, T Lima, P Kersey, M Pagni, CJ Sigrist, C Lachaize, AL Veuthey, E Gasteiger, A Bairoch (2003). Automated annotation of microbial proteomes in SWISS-PROT. *Computational Bio. & Chem.* 27(1):49-58.

The Gene Ontology Consortium (2000). Gene Ontology: tool for the unification of biology. *Nature Genetics* 25:25-29.

RA George, RV Spriggs, GJ Bartlett, A Gutteridge, MW MacArthur, CT Porter, B Al-Lazikani, J Thornton, M Swindells (2005). Effective function annotation through catalytic residue conservation. *Proc. National Acad. Of Sciences* 102(35):12299-12304.

J Goll, R Montgomery, LM Brinkac, S Schobel, DM Harkins, Y Sebastian, S Shrivastava, S Durkin, G Sutton (2009). The Protein Naming Utility: A rules database for protein nomenclature. *Nucleic Acids Res.* 38:D336-D339. doi:10.1093/nar/gkp958.

DH Haft, JD Selengut, O White (2003). The TIGRFAMs database of protein families. *Nucleic Acids Res*. 31:371-373, PMID: 12520025.

DH Haft, JD Selengut, LM Brinkac, N Zafar, O White (2005). Genome Properties: a system for the investigation of prokaryotic genetic content for microbiology, genome annotation and comparative genomics. *Bioinformatics* 21(3):293-306, doi:10.1093/bioinformatics/bti015

S Hunter, R Apweiler, T Attwood, A Bairoch, A Bateman, D Binns, P Bork, U Das, L Daugherty, L Duquenne, R Finn, J Gough, D Haft, N Hulo, D Kahn, E Kelly, A Laugraud, I Letunic, D Lonsdale, R Lopez, M Madera, J Maslen, C McAnulla, J McDowall, J Mistry, A Mitchell, N Mulder, D Natale, C Orengo, A Quinn, J Selengut, C Sigrist, M Thimma, P Thomas, F Valentin, D Wilson, C Wu, & C Yeats (2009). *Nucleic Acids Res.* 37:D211-D215.

Y Ishino, H Okada, M Ikeuchi, H Taniguchi (2007). Mass spectrometry-based prokaryote gene annotation. *Proteomics* 7: 4053-4065.

LJ Jensen, R Gupta, HH Staerfeldt, S Brunak (2003). Prediction of human protein function according to Gene Ontology categories. *Bioinformatics* 2003, 19(5):635-642.

M Kanehisa, S Goto (2000). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res*. 28:27-30.

Peter D. Karp, Suzanne Paley, Pedro Romero (2002). The pathway tools software. *Bioinformatics* 18(Suppl. 1):S225-S232.

RD King, A Karwath, A Clare, L Dehaspe (2001). The utility of different representations of protein sequence for predicting functional class. *Bioinformatics* 17(5):445-454.

W Klimke, R Agarwala, A Badretdin, S Chetvernin, S Ciufo, B Fedorov, B Kiryutin, K O'Neill, W Resch, S Resenchuck, S Schafer, I Tolstoy, T Tatusova (2009). The National Center for Biotechnology Information's protein clusters database. *Nucleic Acids Res*. 37:D216-D223.

E Kretschmann, W Fleischmann, R Apweiler (2001) Automatic rule generation for protein annotation with the C4.5 data mining algorithm applied on Swiss-Prot. *Bioinformatics* 17:920–926.

A Krogh, B Larsson, G von Heijne, EL Sonnhammer (2001). Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. *J. Mol Biol.* 305(3):567-80.

M Kuhn, C von Mering, M Campillos, LJ Jensen, P Bork (2008). STITCH: interaction networks of chemicals and proteins. *Nucleic Acids Res.* 36:D684-8.

E Kuznetsova, Michael Proudfoot, Stephen A Sanders, Jeffrey Reinking, Alexei Savchenko, Cheryl H Arrowsmith, Aled M Edwards, Alexander F Yakunin (2005). Enzyme genomics: Application of general enzymatic screens to discover new enzymes. *FEMS Microbiology Reviews* 29(2): 263-279. doi:10.1016/j.fmrre.2004.12.006.

GRG Lanckriet, M Deng, N Cristianini, MI Jordan, WS Noble (2004). Kernel-based Data Fusion and its Application to Protein Function Prediction in Yeast . *Proceedings of the Pacific Symposium on Biocomputing* 9: 300-311.

Hyunju Lee, Zhidong Tu, Minghua Deng, Fengzhu Sun, Ting Chen (2006). Diffusion Kernel-Based Logistic Regression Models for Protein Function Prediction. *Omics* 10(1): 40-55. doi:10.1089/omi.2006.10.40.

T Lima, A Auchincloss, E Coudert, G Keller, K Michoud, C Rivoire, V Bulliard, E de Castro, C Lachaize, D Baratin, I Phan, L Bougueleret, A Bairoch (2008). HAMAP: A database of completely sequenced microbial proteome sets and manually curated microbial protein families in UniProtKB/Swiss-Prot. *Nucleic Acids Res.* 37(Database): D471-D478.

H Liu, Z-Z Hu, J Zhang, C Wu (2006). BioThesaurus: a web-based thesaurus of protein and gene names. *Bioinformatics* 22(1): 103-105.

B Louie, P Tarczy-Hornoch, R Higdon, E Kolker (2008). Validating annotations for uncharacterized proteins in Shewanella oneidensis. *Omics* 2008(12):211-5.

Ramana Madupu, Lauren M. Brinkac, Jennifer Harrow, Laurens G. Wilming, Ulrike Böhme, Philippe Lamesch, Linda I. Hannick (2010). Meeting report: a workshop on Best Practices in Genome Annotation. *Database* 2010: baq001. doi:10.1093/database/baq001. http://manatee.sourceforge.net/.

Aron Marchler-Bauer, John B. Anderson, Praveen F. Cherukuri, Carol DeWeese-Scott, Lewis Y. Geer, Marc Gwadz, Siqian He, David I. Hurwitz, John D. Jackson, Zhaoxi

Ke, Christopher J. Lanczycki, Cynthia A. Liebert, Chunlei Liu, Fu Lu, Gabriele H. Marchler, Mikhail Mullokandov, Benjamin A. Shoemaker, Vahan Simonyan, James S. Song, Paul A. Thiessen, Roxanne A. Yamashita, Jodie J. Yin, Dachuan Zhang, and Stephen H. Bryant (2005). CDD: a Conserved Domain Database for protein classification. *Nucleic Acids Research* 33(Database): D192-D196.

C von Mering, LJ Jensen, M Kuhn, S CHaffron, T Doerks, B Krūger, B Snel, P Bork (2007). STRING 7 – recent developments in the integration and prediction of protein interactions. *Nucleic Acids Research* 35:D358-D362.

Huaiyu Mi, Betty Lazareva-Ulitsky, Rozina Loo, Anish Kejariwal, Jody Vandergriff, Steven Rabkin, Nan Guo, Anushya Muruganujan, Olivier Doremieux, Michael J. Campbell, Hiroaki Kitano and Paul D. Thomas (2005). The PANTHER database of protein families, subfamilies, functions and pathways. *Nucleic Acids Research* 33:D284-D288.

Huaiyu Mi, Nan Guo, Anish Kejariwal, Paul D. Thomas (2007). PANTHER version 6: protein sequence and function evolution data with expanded representation of biological pathways. *Nucleic Acids Res.* 35:D247–D252.

S Mostafavi S, D Ray, D Warde-Farley, C Grouios, Q Morris (2008) GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biology* 9:S4. doi:10.1186/gb-2008-9-s1-s4.

D Natale, C Arighi, WC Barker, J Blake, T-C Chang, Z Hu, H Liu, B Smith, CH Wu (2007). Framework for a protein ontology. *BMC Bioinformatics* 8(Suppl 9):S1.

R Overbeek, T Begley, RM Butler RM, JV Choudhuri, HY Chuang, M Cohoon M, V de Crecy-Lagard, N Diaz, EM Glass, A Goesmann, A Hanson, D Iwata-Reuyl, R Jensen, N Jamshidi, L Krause, M Kubal, N Larsen, B Linke, AC McHardy, F Meyer, H Neuweger, G Olsen, R Olson, A Osterman, V Portnoy, GD Pusch, DA Rodionov, C Rückert, J Steiner, R Stevens, I Thiele, O Vassieva, Y Ye, O Zagnitko, V Vonstein (2005). The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic Acids Research* 13(17): 5691-5702.

P Pavlidis, J Weston, J Cai, WS Noble (2002). Learning gene functional classifications from multiple data types. *J Comp Bio* 9: 401-411.

L Peña-Castillo, M Tasan, CL Myers, H Lee, T Josh, C Zhang, Y Guan, M Leone, A Pagnani, WK Kim, C Krumpelman, W Tian, G Obozinski, Y Qi, S Mostafavi, GN Lin, GF Berriz, FD Gibbons, G Lanckriet, J Qiu, C Grant, Z Barutcuoglu, DP Hill, D Warde-Farley, C Grouios, D Ray, JA Blacke, M Deng, MI Jordan, WS Noble, Q Morris, J Klein-Seetharaman, Z Bar-Joseph, T Chen, F Sun, OG Troyanskaya, EM Marcotte, D Xu, TR Hughes, FP Roth (2008). A Critical Assessment of M. Musculus

Gene Function Prediction using Integrated Genomic Evidence. *Genome Biology* 9(Suppl.1):S2.

M Riley, DB Space (1995). Genes and proteins of *Escherichia coli* (GenProtEc). *Nucleic Acids Research* 24(1):40.

V Sangar, DJ Blankenberg, N Altman, AM Lesk (2007). Quantitative sequence-function relationships in proteins based on gene ontology. *BMC Bioinformatics* 8:294. doi:10.1186/1471-2105-8-294.

LF Schroeder, ALC Bazzan, JF Valiati, PM Engel, S Ceroni (2002). A comparison between symbolic and non-symbolic machine learning techniques in automated annotation of the "Keywords" field of SWISS-PROT. In Gramado, RS. Proc. of the First Brazilian Workshop on Bioinformatics. Soc. Bras. Computação.

Jeremy Selengut, Daniel Haft, Tanja Davidsen, A. Ganapathy, M. Gwinn-Giglio, William Nelson, Alex Richter, Owen White (2007). TIGRFAMs and Genome Properties: tools for the assignment of molecular function and biological process in prokaryotic genomes. *Nucleic Acids Res.* 35:D260-4.

B Shahbaba, RM Neal RM (2006). Gene function classification using Bayesian models with hierarchy-based priors. *BMC Bioinfo* 7:448.

Roden Sharan, Igor Ulitsky, Ron Shamir (2007). Network-based prediction of protein function. *Molecular Systems Biology* 3(88): 1-13.

CJA Sigrist, L Cerutti, N Hulo, A Gattiker, L Falquet, M Pagni, A Bairoch, P Bucher (2002). PROSITE: a documented database using patterns and profiles as motif descriptors. *Briefings in Bioinform.* 3:265-274.

T Slater, C Bouton, ES. Huang (2008). Beyond data integration. *Drug Discovery Today* 13: p. 584-589.

TF Smith, MS Waterman (1981). Identification of Common Molecular Subsequences. *Journal of Molecular Biology* 147: 195–197.

Roman L. Tatusov, Michael Y. Galperin, Darren A. Natale, and Eugene V. Koonin (2000). The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Res.* 28(1): 33–36.

RM Ward, S Erdin, TA Tran, DM Kristensen, AM Lisewski (2008). De-Orphaning the Structural Proteome through Reciprocal Comparison of Evolutionarily Important Structural Features. *PLoS ONE* 3(5): e2136 doi:10.1371/journal.pone.0002136

O Whelehan, M Earll, E Johansson, M Toft, L Eriksson (2006). Detection of ovarian cancer using chemometric analysis of proteomic profiles. Chemometrics and intelligent Laboratory System 84 (1/2):82-87

D Wieser, E Kretschmann, R Apweiler (2004) Filtering erroneous protein annotation. *Bioinformatics*, 20, i342-i347.

C Wu, A Nikolskaya, H Huang, L-S Yeh, D Natale, CR Vinayaka, Z-Z Hu, R Mazumder, S Kumar, P Kourtesis, RS Ledley, BE Suzek, L Arminski, Y Chen, J Zhang, JL Cardenas, S Chung, J Castro-Alvear, G Dinkov, W Barker (2004). PIRSF: Family classification system at the Protein Information Resource. *Nucleic Acids Res.* 2004 32(Database): D112-D114.

C Yu, N Zavaljevski, V Desai, S Johnson, FJ Stevens, J Reifman (2008). The development of PIPA: an integrated and automated pipeline for genome-wide protein function annotation. *BMC Bioinformatics* 9: 52.

GX Yu (2004) RuleMiner: a knowledge system for supporting high-throughput protein function annotations. *J of Bioinformatics and Comp Bio* 2:595-617.

Haiyuan Yu, Pascal Braun, Muhammed A. Yildirim, Irma Lemmens, Kavitha Venkatesan, Julie Sahalie, Tomoko Hirozane-Kishikawa, Fana Gebreab, Na Li, Nicolas Simonis, Tong Hao, Jean-François Rual, Amélie Dricot, Alexei Vazquez, Ryan R. Murray, Christophe Simon, Leah Tardivo, Stanley Tam, Nenad Svrzikapa, Changyu Fan, Anne-Sophie de Smet, Adriana Motyl, Michael E. Hudson, Juyong Park, Xiaofeng Xin, Michael E. Cusick, Troy Moore, Charlie Boone, Michael Snyder, Frederick P. Roth, Albert-László Barabási, Jan Tavernier, David E. Hill, and Marc Vidal (2008). High-Quality Binary Protein Interaction Map of the Yeast Interactome Network. *Science* 3 October 2008: 104-110.

Curriculum Vitae

Philip Goetz attended Loyola College in Maryland, receiving a Bachelor of Science in Mathematics with a minor in Writing in 1989. He received a Doctorate in Computer Science, with a minor in Linguistics, from the State University of New York at Buffalo in 1997. He has done work for the National Aeronautics and Space Administration, the U.S. Army Advanced Research Labs, the Defense Advanced Research Projects Agency, and the National Institutes of Health. He currently works for the J. Craig Venter Institute, where he performed the work described in this thesis.