

Automated learning of protein involvement in pathogenesis using
integrated queries

Eithon Cadag

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2009

Program Authorized to Offer Degree: Department of Medical Education and Biomedical
Informatics

UMI Number: 3394276

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

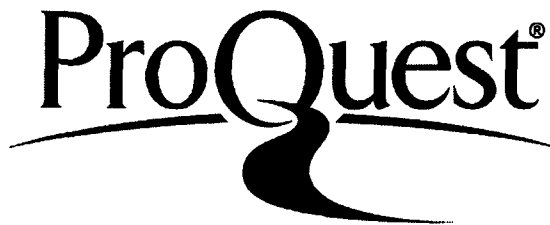
In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3394276

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

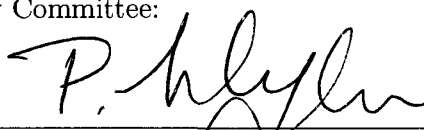
University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Eithon Cadag

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:



Peter J. Myler

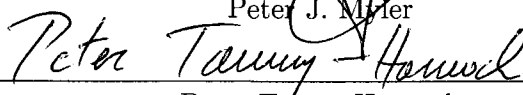
Reading Committee:



Ira J. Kalet



Peter J. Myler

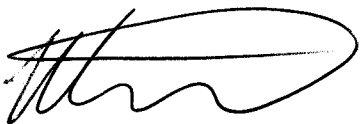


Peter Tarczy-Hornoch

Date:

12/9/09

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature  _____

Date 12/15/09 _____

University of Washington

Abstract

Automated learning of protein involvement in pathogenesis using integrated queries

Eithon Cadag

Chair of the Supervisory Committee:
Research Professor Peter J. Myler
Department of Medical Education & Biomedical Informatics,
and Department of Global Health

Methods of weakening and attenuating pathogens' abilities to infect and propagate in a host, and thus allowing the natural immune system to more easily decimate invaders, have gained attention as alternatives to broad-spectrum targeting approaches. The following work describes a technique to identifying proteins involved in virulence by relying on latent information computationally gathered across biological repositories. A lightweight method for data integration is introduced, which links information regarding a protein via a path-based query graph and supports both exploratory and logical queries; data gathered in this way is characterized with experiments on retrieving high-quality annotation data. A system and method of weighting is then applied to query graphs that can serve as input to various statistical classification methods for discrimination, and the combined usage of both data integration and learning methods are leveraged against the problem of generalized and specific virulence function prediction. This approach improves coverage of functional data over a protein, outperforms other recent approaches to identification of virulence factors, is robust to different weighting schemes of varying complexity and is found to generalize well to traditional function prediction.

TABLE OF CONTENTS

	Page
List of Figures	iv
List of Tables	ix
Chapter 1: Introduction	1
1.1 Dissertation progression	2
1.2 Novel contributions	3
1.3 Relevant biology	5
1.3.1 Sequence and structure	5
1.3.2 Protein functional analysis	5
Chapter 2: Challenges in Identifying and Elucidating Pathogenic Proteins	8
2.1 Microbial pathogens and virulence factors	8
2.1.1 Pathogen targets and virulence factors	10
2.1.2 Challenges to identifying virulence factors	12
2.1.3 Computational methods for finding virulence factors	13
2.2 Protein identification and annotation	15
2.2.1 Manual and automated methods	15
2.2.2 Challenges to automated annotation	16
2.3 Strengths and weaknesses of identification and annotation methods	18
Chapter 3: Methods of Data Integration and Statistical Classification in Biology	19
3.1 Computationally predicting protein function	19
3.1.1 Supervised learning	19
3.1.2 Review of learning methods in biology	20
3.2 Data integration methods in biology	25
3.2.1 Federated data integration	28
3.2.2 Path-based federation	31
3.3 Discussion	33

Chapter 4:	Model and Implementation of a Biologic Data Integration Engine . . .	35
4.1	System architecture	36
4.1.1	Frame-based mediated schema	38
4.1.2	Query graph and browsing engine	41
4.1.3	Source interfaces	44
4.1.4	Support for structured queries	45
4.2	Motivations for design choices	48
4.3	Discussion	51
Chapter 5:	Characterizing Integrated Access to Biological Information	52
5.1	Reachability of trustworthy integrated data	53
5.2	Related work	54
5.3	Methods	54
5.3.1	Fault tolerance model over arbitrary query graphs	54
5.3.2	Estimating tolerance of biological queries for high-quality data	57
5.4	Results	60
5.5	Conclusion	66
5.6	Discussion	67
Chapter 6:	Learning Pathogenic Proteins from Integrated Query Networks	69
6.1	Methods	69
6.1.1	Learning query graphs	69
6.1.2	Evaluation methods	73
6.2	Results	75
6.2.1	Query results characteristics	75
6.2.2	Performance across sources, learning methods	78
6.2.3	Comparison of query graph weighting schemes	82
6.3	Conclusion	82
6.4	Discussion	85
Chapter 7:	Learning Protein Involvement in Specific Pathogenic Roles	86
7.1	Methods	86
7.1.1	Dataset preparation	86
7.1.2	Curating virulence factor subclasses	87
7.1.3	Baseline classifiers	90
7.1.4	Evaluation methods	93

7.2	Results	95
7.2.1	Coverage and summary statistics	95
7.2.2	Source-against-source performance	98
7.2.3	Data integrated learning vs. baseline methods	100
7.3	Conclusion	106
7.4	Discussion	108
Chapter 8:	Generalizability of Query- and Data-Level Integration and Learning	110
8.1	Applicability of methods and concepts	110
8.2	Case study: predicting generalized protein function	110
8.2.1	Methods	111
8.2.2	Results	113
8.2.3	Discussion	115
Chapter 9:	Contributions and Significance of Work	117
9.1	Novelty of path-based federated integration and learning	117
9.2	Novel biological implications	121
9.3	Limitations of methods and results	123
9.4	Future work	125
9.5	Concluding remarks	127
Appendix A:	Integration Architecture	129
A.1	Query graph definitions	129
A.2	DaRQL grammar	148
A.3	DaRQL query planning benchmarks	149
Appendix B:	Learning experiments	150
B.1	Characteristics of the query graph	150
B.2	Generalized virulence results	151
B.3	Statistical significance of specific virulence	152
Bibliography	158

LIST OF FIGURES

Figure Number	Page	
1.1	In the above, 1.1a is the genomic sequence that encodes the protein in 1.1b, which itself is one half of the corresponding homodimer shown in 1.1c (generated using [1]). The protein, <i>ESXa</i> , is implicated in host invasion and is associated with a <i>S. aureus</i> secretory pathway [2].	6
1.2	General workflow process of functional determination from sequence; ‘Computational methods’ refers to the reliance of primarily local or global sequence-based homology, while ‘Experimental methods’ refers to structural elucidation or mutation and knockout studies.	7
2.1	Infectious disease incidences across the globe from July 22 to August 20, 2009, colorized by urgency (deeper color is more urgent); this image was generated from [3] using data from the World Health Organization, ProMED [4, 5] and Eurosurveillance [6]. Areas of dense population, overuse of antibiotics and zoonosis tend to dominate instances of emerging infections disease [7].	9
3.1	Above, Fig. 3.1a shows a series of positive (red circles) and negative (black crosses) instances which are not linearly separable without large error in their native 2D space; Fig. 3.1b shows the same data plotted with a mapping (Φ) to a 3D space (with $z = (xy - \frac{x}{2} - \frac{y}{2} + \frac{1}{4})^2$) where a linear separation is possible with less error.	23
3.2	The linearly-increasing number of biological databases from 2000 to 2009 [8, 9, 10, 11, 12, 13, 14, 15, 16]; it would be physically near-impossible for a single scientist to manually parse through nearly up to 1200 databases in search of relevant information in a timely fashion.	27
3.3	Graphical representation of the query expansion for $q \leftarrow gene(a)$	32
4.1	Architecture of the core MIQAPS system, where arrows denote information flow between the various components (dotted indicates a non-required interaction). Queries originate on the left (q) and are posed to a <i>query graph</i> instance either in an exploratory (key-value) fashion or using a path-based structured query language.	37
4.2	Formal entity declaration in the MIQAPS mediated schema in extended Backus-Naur form. The ‘.’ symbol denotes an attribute of single arity, while ‘*’ indicates an attribute of n -nary arity.	39

4.3	Formal source definition in the MIQAPS mediated schema in extended Backus-Naur form.	40
4.4	Formal link declaration in the MIQAPS mediated schema in extended Backus-Naur form.	41
4.5	The above shows the data sources (in grey, dotted boxes), associated classes (ellipses), cross-links (arrows) and cardinalities (arrow labels) for the integrated data retrieval system. Queries begin on the left, with the <i>Seed</i> source, and are expanded rightward.	42
4.6	A small query graph generated using MIQAPS. Query graphs generated using MIQAPS are directed and acyclic.	43
4.7	Basic structure of a DaRQL query.	46
4.8	Example query in DaRQL.	47
5.1	Sampling procedure used to approximate $p(G, R, l)$ with respect to target nodes R , in query graph G , at a failure rate l ; applying this with $l = (0.0, \dots, 1.0)$ yields $\tau \approx \frac{1}{ l } \cdot \sum_{k \in l} p(G, R, k)$	56
5.2	MIQAPS architecture, adapted for fault tolerance experiments. The <i>tolerance module</i> enacts the simulation over saved query graphs generated using exploratory sequence queries.	58
5.3	Based on 11 randomly selected queries, convergence of the fault tolerance measure τ started at approximately 10 000 simulations.	59
5.4	Selected archetypal tolerance curves (τ_{exp}) for experimentally validated Gene Ontology terms compared to non-experimental (τ_{rdm}).	61
5.5	Histogram of $\tau_{exp} - \tau_{rdm}$ for the test proteins. Values below 0 indicate that random GO terms are more fault tolerant to node loss, and thus more reachable, than experimental GO terms, and <i>vice versa</i>	62
5.6	Average tolerances and standard deviations between experimental GO terms and random GO terms.	63
5.7	Comparison of the random tolerance curve from query graphs, to the tolerance of similarly-composed power law and Erdős-Rényi graphs. Above, $\tau_{rdm} = 0.601$, $\tau_{pow} = 0.553$ and $\tau_{erg} = 0.406$	66
6.1	Adapted schema used for virulence identification, with CDD, GENNAV and UNIPROT added. Note that GENNAV is a recursive source – that is, it may re-query itself to recreate the GO hierarchy within the query graph.	74
6.2	In the above, Fig. 6.2a is the frequency of the in-degrees of nodes in all 2055 query graphs (with detailed inset), and Fig. 6.2b is a log-log plot of the in-degree cumulative probability distribution; the seed nodes are omitted in these charts.	77

6.3	Frequency plots of the propagation weights across non-virulent and virulent proteins, by source; Fig. 6.3i shows the periodic influence of λ on discretizing records beyond the query node.	79
6.4	ROC curves for the eight different data sources based on the test data. Parameters were coarsely optimized for the AUC.	80
6.5	Plot of ROC scores between using propagation-weighted nodes (y -axis) and binary-weighted nodes (x -axis); done using SVM-based classification.	83
7.1	Inter- and intra-category similarity distribution by identity, pre-40% identity clustering and using <code>blastall</code> with the low-complexity filter off.	92
7.2	Feature distributions by source on log-y plots; the x-axis refers to individual features, sorted descending by the number of instances in which they appear (in the y-axis). Less precipitous drops indicate sources where the appearance of any given feature in any instance is more evenly distributed.	97
7.3	ROC curves for data-integrated sources using optimized parameters.	99
7.4	Statistical significances based on six five-fold cross-validations for all 11 virulence classes. An arrow from a head source or method to a tail source or method transitively indicates better pairwise performance from the head against the tail. Results, <i>via</i> two-tailed t -test, for ROC and ROC ₅₀ are shown and nodes are color-coded based on source type (blue indicates domain- or motif-based sources, red GO term-based sources, yellow-brown for pathways, gray for structural sources and green for baseline methods). Actual values are listed in Tbls. B.1- B.13.	101
7.5	Fig. 7.5a shows histograms of query graph sizes for a subset of instances; those within PDB's ROC ₅₀ curve ('PDB top-scoring') were generally larger than those outside of it ('PDB low-scoring'). The difference between the two distributions (which appear shifted, though symmetrical) is statistically significant with a p -value less than $2.2e-16$, <i>via</i> two-tailed signed rank test. Fig. 7.5b shows cumulative probability distributions for the number of instances within the ROC ₅₀ of PDB and GENNAV against the baseline of all query graphs. PDB instances that fall under the ROC ₅₀ tend to be larger than both the overall average and GENNAV. These charts omit query graphs for which no results were returned.	102
7.6	Average AUCs and 95% confidence intervals for a subset of sources and the baselines by training set size, based on three five-fold cross-validations. The 'Surface factor' virulence class is omitted due to the small number of instances present in the training set.	104

7.7	Improvement in AUC of using an SVM for classification for cross-validated tests over $Top-F_1$. The colors in each cell correspond to the difference in AUC between using an SVM for prediction and using the top-performing feature per source, per label. Higher values indicate cases where there is added benefit in considering multiple features <i>via</i> SVMs.	106
8.1	Inter- and intra-category similarity distribution by identity.	112
8.2	Comparison of ROC performance on 13 MIPS functional groups using Markov random fields (MRF), fused kernels with SVMs enriched with Pfam data (SDP+SVM), integrated weight-averaging (IWA), BLAST-based k NN, integrated query graphs with SVMs (DI+SVM) and integrated query graphs and $3mer$ in a fused kernel (DI+ikSVM). Error bars for denote 95% confidence intervals for average ROC, based on 15 cross-validated experiments. The data for MRF and SDP+SVM are from [17], and the data for IWA are from [18].	115
9.1	Perceived comparison of three path-based federated approaches (A -discriminative, the method of this dissertation; B -logical, as described in [19]; C -belief, as described in [20, 21]) to deriving biological knowledge along three axes (cost, quality and scalability). These methods are juxtaposed against the backdrop of the cost, quality and scalability levels traditionally associated with semi-automated and automated methods, manual curation methods and methods that produce experimentally validated evidence.	120
A.1	DaRQL production rules, in extended Backus-Naur Form.	148
A.2	Query execution times for 99 randomly selected proteins from yeast, without query planning and with query planning for a restrictive query with many constraints and for a loosely-defined query with few constraints; solid horizontal lines mark the means for each query, and dotted horizontal lines the standard deviations. Benchmarks were done on an Intel dual core 2.66GHz Linux machine with 3GB of RAM.	149
B.1	A power-law model fitted to the distribution frequency of in-degrees in the query graphs (Fig. B.1a), and the residuals (Fig. B.1b). The model was fitted <i>via</i> unweighted least-squares regression using the formula $P_k = k^{-\alpha}$, with P_k the distribution frequency for in-degree k . α was estimated to be 1.67.	150
B.2	Individual propagation weights for GO terms, based on virulent and non-virulent records; Fig. B.2b is a zoomed-in subset of Fig. B.2a.	151

B.3 Progressive results of integrating kernel sources *via* (3.7) into a sequence-baseline (a) SVM classifier (source order is (a+) GENNAV (\rightarrow b), (a + b+) AMIGO (\rightarrow c), INTERPRO (d), KEGG (e), CDD (f)). There was no significant improvement in using integrated kernels using equal weighting; *ad hoc* weighting of the kernels did show some improvement (not shown), but weight-tuning was not done comprehensively. 152

LIST OF TABLES

Table Number	Page
2.1	Abbreviated virulence classification proposed by Wassenaar and Gaastra; the complete classification system is available from [22]. The working definition of virulence factor used in this dissertation encompasses all of the above. . . . 11
2.2	Sequence databases curated specifically for virulence factors, area of specialty and relevant dates of first publication; note that this list is not meant to be comprehensive of all virulence-specific sequence databases [23, 24, 25, 26, 27, 28, 29, 30, 31]. 14
2.3	The state of annotation and gene identification for the human genome, model genomes and a number of protozoan genomes of public health interest [32]. Notably, the proteomes of many infectious disease organisms remain functionally unsolved. 17
4.1	A comparison of biological data integration systems along selected features. An asterisk (*) indicates that the integration architecture supports path-based queries. <i>Guided queries</i> refer to systems with sufficiently expressive query languages that the query path may be parameterized; <i>user-centric queries</i> refer to systems geared toward exploratory queries; and systems denoted with <i>infrastructure</i> are those that are explicitly designed to be layered with other tools, and are not necessarily stand-alone <i>per se</i> [33, 34, 35, 36, 37, 38, 39, 40, 41]. 50
5.1	Experimental Gene Ontology evidence codes, from http://www.geneontology.org 59
5.2	Top 10 and bottom 10 in $\tau_{exp-rdm}$; proteins where the experimental GO terms are likely to stand out are from mammalian proteomes, or have been reviewed. 64
5.3	Average query graph tolerances for experimental Gene Ontology terms with source-targeted node failure, and likelihoods that a term is experimentally derived, given that a source has a path to it. 65
6.1	Database coverage by fraction across all sources for the training and test sets by MIQAPS. Source sizes are estimates as of January 2009. 76

6.2	Results by source and method for predicting virulent and non-virulent bacterial proteins given AUC. The best performer, GENNAV was run with a Gaussian kernel whose $\sigma = 1.0$ and regularization cost $C = 1.0$	78
6.3	Above are results by source, when empty graphs (query graphs with no returned results) are excluded from training and testing; the scores are thus those of each source given data from that source was available.	81
6.4	Comparison of the top-performing integrated predictor against a sequence baseline and VirulentPred; all methods outlined in the table used the same set of proteins.	82
7.1	Iterative method used to manually align and annotate the virulence classifications for pathogenic proteins in the training and testing dataset.	89
7.2	11 top-level virulence factor categories derived from the positive training and testing set, with subclassifications. Importantly, a virulence factor may be classified under several labels, <i>e.g.</i> , a protein may be both a ‘Surface factor’ (label 2) and an ‘Antigen’ (label 10).	91
7.3	The 11 main virulence categories derived manually from the pathogenic protein data sources with the number of training and testing records, after 40% identity pruning.	95
7.4	Database coverage by fraction across all sources for the training and test sets by MIQAPS for the specific virulence dataset.	96
8.1	MIPS functional categories, with protein counts drawn from yeast.	113
8.2	Database coverage by fraction across all sources for the training and test sets by MIQAPS for the MIPS protein set.	114
9.1	Tbl. 9.1a shows the number of predicted virulence proteins in <i>B. pseudomallei</i> , via data integration and SVM learning. GENNAV GO terms were used as the predictive features, and the threshold was set by 95% precision of the specific virulence dataset. Tbl. 9.1b is a sampling of the top-ranked possible virulence factors from the predictions.	122
B.1	ROC p-values for the Adherence virulent class.	153
B.2	ROC p-values for the Surface factor virulent class.	153
B.3	ROC p-values for the Invasion virulent class.	154
B.4	ROC p-values for the Transport and uptake virulent class.	154
B.5	ROC p-values for the Toxin virulent class.	154
B.6	ROC p-values for the Catalysis virulent class.	155
B.7	ROC p-values for the Secretion virulent class.	155
B.8	ROC p-values for the Motility virulent class.	155
B.9	ROC p-values for the Antibiotic resistance virulent class.	156

B.10 ROC p-values for the Defense virulent class.	156
B.11 ROC p-values for the Other virulent class.	156
B.12 ROC means across specific virulence classes, derived from six five-fold cross- validations (corresponding to the significance tests).	157
B.13 ROC ₅₀ means across specific virulence classes, derived from six five-fold cross- validations.	157

ACKNOWLEDGMENTS

I would like to thank the following organizations, without whose financial support this research would not have been possible: the National Human Genome Research Institute (grant R01HG02288), the National Library of Medicine (grant T15LM07442), the National Institute of Allergy and Infectious Diseases (contract HHSN 272200700057 C) and the U.S. Department of Defense (grant N00244-09-1-0081).

I want to recognize and thank the faculty and staff of UW Biomedical and Health Informatics and Seattle Biomedical Research Institute for their support all these years; folks who have provided significant aid, advice or encouragement during my time as a graduate student include: Brian Brown, Sandy Turner, Joan San, Ron Shaker and Janos Barbero at the UW, and Andrew Leonard at SBRI. I would also like to acknowledge my fellow students in Myler and Bio-DIAG labs and UW BHI for their company and friendship, and in particular Dhileep Sivam, Terry Shen, Sophia Jeng and Brent Louie.

My work has greatly benefited from the guidance of my doctoral committee: Evan E. Eichler, Ira J. Kalet, William S. Noble, with special gratitude to Peter Tarczy-Hornoch, and my advisor, Peter J. Myler. I am indebted to these individuals not only for the thoughtful and constructive advice they have so freely provided all these years, but also for serving as my role models for scholarly and scientific activity.

Last but not least, I would like to express how grateful I am for my supportive family and friends, especially Mom, Dad, Ian and Betty – I love you guys!

Eithon Cadag

Chapter 1

INTRODUCTION

Recorded history is replete with stories of mankind's eternal struggle against infectious disease. From the plague of Athens in 430 B.C., the Black Death of Europe in the 14th century, the Spanish flu of 1918 to the more modern pandemic of HIV/AIDS, bacterial and viral agents have been and remain to be responsible for millions of deaths across the globe. Indeed, in 2008 infectious diseases were attributed to approximately 10 million deaths [42], a conservative estimate that does not take into account our limited understanding of the role they may play in chronic conditions such as cancer or heart disease [43, 44]. Despite significant and admirable advances in both biomedical research and clinical care around infectious disease, emergent and re-emergent infectious diseases, antibiotic-resistant microbes and weaponization of infectious agents are compelling arguments for continuing research and development in methods leading to effective therapeutic countermeasures.

Though there are numerous obstacles in bringing basic research to clinical and applied fruition, the first challenge that many biomedical scientists often face when beginning to analyze their data and prioritize research attention is understanding what role each individual gene, protein or pathway plays. The initial step in this process is the integration of data across biological repositories [45, 46], the importance of which cannot be understated given that many of these data sources collectively contain a wealth of information that can be quickly and cheaply gathered in comparison to traditional wetlab benchwork. A major computational challenge in the post-genomic era is thus how best to query, mine and exploit the myriad biological information being generated and published; as resources are limited and therefore allocated to studies and experiments that could potentially have the most beneficial impact on human health, effective analysis and timely prioritization can help accelerate the drug development process.

This dissertation introduces a method and system that takes advantage of integrated

data from disparate biological sources to identify proteins in infectious organisms that may be involved in virulence within human hosts. The approach is further demonstrated to be applicable beyond pathogenicity and is capable of predicting general protein function. The crux of this method is the data linkages afforded by path-based querying over cross-referencing data sources, and the advantage a formal data integration implementation provides in coverage for any given protein in comparison to singular protein queries posed to databases individually. Additional benefit is added when the query results are enriched with weight scores generated iteratively for each individual result. By relying on query-level data integration, proteins are mapped into a space whose composition is defined by numerous data sources; these mappings are trained upon by a statistical classifier, and in this way arbitrary protein classifications can be learned.

1.1 *Dissertation progression*

The following dissertation proposes a novel approach to combining data and statistical learning methods *via* integration at the query level, and tests this approach on identification and ranking of virulence proteins across a variety of pathogenic roles. This chapter enumerates the contributions this coupling makes to areas of bioinformatics, data management and biological data mining, in the form of artifacts, methods and experimental findings. As the biological data integration and statistical learning approaches are both applied generally to the task of identification, the end of this chapter is marked with a brief review of and background information on gene and protein annotation.

Immediately following this chapter, Chapter 2 discusses in detail the problem of virulence annotation and challenges related to adequate prioritization of proteins involved in disease pathogenesis. Chapter 2 further explores methods of manual annotation for virulence detection, and outlines deficiencies in enabling more comprehensive and rapid understanding of infectious disease processes.

Chapter 3 formally defines both statistical learning and data integration over biological data sources, and provides a literature review of approaches from both fields used for protein annotation of virulence factors and generalized protein function. The chapter concludes with a description of a method and implementation that uses *path-based integrated queries*

across multiple, heterogeneous data sources as a means of gathering information of varying relevance concerning a query. Chapter 5 describes an experiment to test the ability of this approach to retrieve high-quality biological information in the face of noisy data, and shows that naïve integration alone is insufficient to derive information without encountering high rates of annotation error.

Next, Chapter 6 describes a method capable of transforming data in path-based integrated queries into a form amenable for machine learning. Using a curated dataset of virulent and non-virulent proteins, it is shown that a combination of graph-based query-level integration and learning methods outperforms other methods for generalized virulence prediction.

Chapter 7 builds on the experiments of Ch. 6 by applying similar experiments to the problem of specific virulence factor identification – that is, the identification of exact roles proteins may play in infectious disease. The chapter outlines the process of manually curating a dataset organized by specific virulence protein labels, and describes the results of using integrated query graphs and support vector machine-based learning techniques to rank proteins by their involvement in specific pathogenic processes. These methods are generalized in Chapter 8, which reports results generated from using integrated queries and learning for overall function prediction. This combined approach shows further promise, and outperforms prior reported results from other studies and baseline methods.

1.2 Novel contributions

Both data integration methods and machine learning techniques have been applied to the domain of biology in the past, and many computational approaches for querying, managing and mining data have been adopted within biology. Data integration is a common approach to querying and retrieval; statistical learning is often used for biological classification and ranking, and more increasingly involves methods that incorporate integrated data. It is the hope that this dissertation’s contribution is to broadly show the effectiveness of combining both these practices formally to take advantages of the strengths and ameliorate the weaknesses each may have alone, thereby showcasing the efficacy for learning that integration of data at the *query* level affords. Specifically, this dissertation makes the following

contributions:

1. A lightweight data integration system and schema developed for protein virulence identification that cross-links data into an integrated query graph, as well as a novel query language for supporting ambiguous and ranged queries over integrated biomedical data.
2. Experiments to show the advantage of coverage and redundancy such a system provides. These experiments also suggest that data integration by itself may not be enough to separate high-quality data from poorer-quality data.
3. A process and method of adapting integrated query graphs generated from the data integration system into a vector form amenable to machine learning. This process allows any query to be mapped into a space defined by constituent data sources, and thus each integrated query graph can become a canonical exemplar of a query. When combined with statistical learning techniques, integrated query graphs become learnable instances.
4. Experiments to show that this method of query result transformation is capable of learning generalized virulence, and performs better than prior methods of generalized virulence identification and ranking.
5. Further experiments to show that combining integrated queries and machine learning can be applied effectively to the more difficult task of specific virulence, where it outperforms baseline classification methods.
6. Experiments to show that integrated queries can be used for the broad problem of generalized protein function, where it is compared to prior methods that partly rely on experimental data with favorable results.

The above contributions demonstrate the usefulness of coupling formal data integration techniques with learning methods for a field where gene prioritization and relevancy ranking are an important step in the discovery pipeline.

1.3 Relevant biology

The following section contains a brief introduction to aspects of molecular biology that are pertinent to this dissertation; a more comprehensive review of this topic in the context of computational research is available from other sources [47, 48].

1.3.1 Sequence and structure

Many modern computational sequence analysis methods rely on the *central dogma* of genetics as a basis – that is, information encoded in DNA (deoxyribonucleic acid) is transcribed from the genome into RNA (ribonucleic acid), which in turn is used during translation of molecules composed of amino acids. DNA, composed of the four-letter nucleic acid alphabet $\Lambda_{DNA} = \{A, C, T, G\}$, is organized in the genome as *genes*; the translated molecules generally are composed from a larger 20-letter amino acid alphabet which, when complete, form a *protein* with three-dimensional structure and motion. This information transfer from DNA to protein can be characterized as direct, with a correspondence of three nucleic acids to one amino acid. Once translated, proteins use interaction with other molecules to carry out one or more functions within, or in some cases, externally from (see Fig. 1.1), the cell; the conformation of the protein is integral in carrying out its tasks, and generally function follows from form.

1.3.2 Protein functional analysis

Unfortunately, structural elucidation of proteins *via* laboratory work can be very difficult, time-intensive and costly using traditional X-ray crystallography and nuclear magnetic resonance (NMR) methods [49, 50, 51]. Largely due to advances in computational hardware and algorithms a much faster, albeit more uncertain, approach is to infer the function of a protein from its *sequence homology* to other, known proteins – that is, using a variety of statistical measures and given a protein of unknown function or role, identify its purpose by examining how similar its sequence is to other sequences whose protein function is known.

Using this leap from sequence to function, search algorithms such as BLAST and its variants [52, 53] can compare a protein against a database of millions of sequences within

```

ATGGCAATGATTAAGATGAGTCCAGAGGAAATCAGAGCAAATCGCAATCTTACGGGCAA
GGTTCAGACCAAATCCGTCAAATTTTATCTGATTTAACACGTGCACAAGGTGAAATTGCA
GCGAACTGGGAAGGTCAAGCTTTCAGCCGTTTCGAAGAGCAATTCCAACAACCTTAGTCCT
AAAGTAGAAAAATTTGCACAATTATTAGAAGAAATTAACAACAATTGAATAGCACTGCT
GATGCCGTTCAAGAGCAAGACCAACAACCTTCTAATAATTTTCGGTTTGCAATAA

```

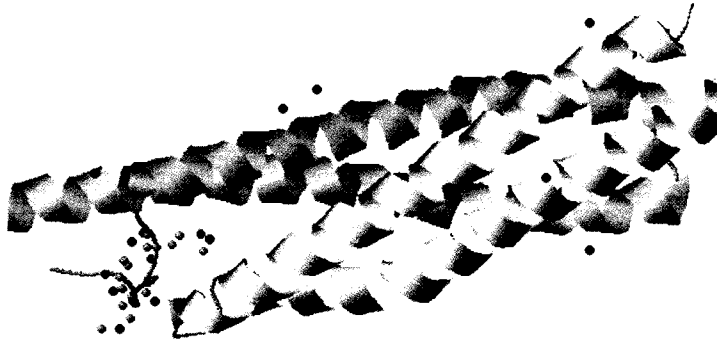
(a)

```

MAMIKMSPEEIRAKSQSYGGSDQIRQLSDLTRAQGEIAANWEGQAFSRFEEQFQQLSP
KVEKFAQLLEEIKQLNSTADAVQEQQQLSNNFGLQ

```

(b)



(c)

Figure 1.1: In the above, 1.1a is the genomic sequence that encodes the protein in 1.1b, which itself is one half of the corresponding homodimer shown in 1.1c (generated using [1]). The protein, *ESXa*, is implicated in host invasion and is associated with a *S. aureus* secretory pathway [2].

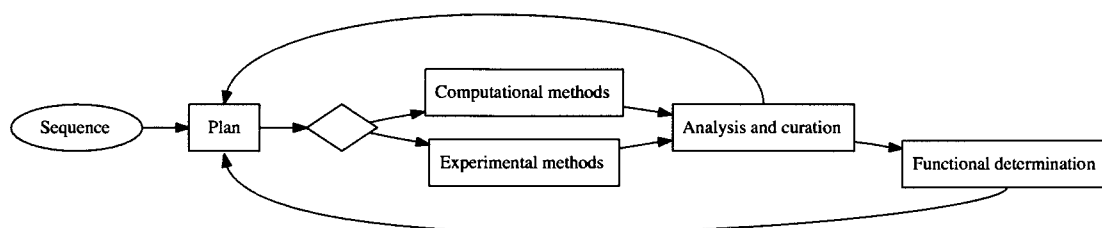


Figure 1.2: General workflow process of functional determination from sequence; ‘Computational methods’ refers to the reliance of primarily local or global sequence-based homology, while ‘Experimental methods’ refers to structural elucidation or mutation and knockout studies.

minutes and with negligible cost and effort when compared to lab-based functional analysis. Further analyses to corroborate findings could include remote homology detection, transmembrane region prediction, cellular localization and computational prediction of secondary and tertiary protein structure [54]. For example, *hidden Markov models* (HMMs) are commonly used to detect distant homologies, where a sequence may dramatically differ from the query despite sharing similar structure [55, 56]. An *annotator* would then depend on their knowledge to select the most accurate and characterizing function or set of functions to assign to their unknown protein (see also Ch. 2.2 for a deeper discussion on protein annotation).

Computational approaches thus provide an inexpensive and fast way of assigning preliminary annotations to proteins, and can help scientists prioritize proteins of initial interest. Ideally, computational function assignment would be followed by, or conducted in parallel with, more rigorous experimental assays to confirm findings, as shown in Fig. 1.2. Furthermore, curation and functional determination is often an incremental process that needs continual maintenance to remain up-to-date, as new information is discovered or old information corrected.

Chapter 2

**CHALLENGES IN IDENTIFYING AND ELUCIDATING
PATHOGENIC PROTEINS**

Among the first steps to understanding the mechanisms of infectious disease is the identification and annotation of the specific proteins involved in pathogenesis. This chapter underlines the importance of infectious disease research and provides a definition of *virulence factors*, those proteins involved in invasion and maintenance of disease-causing microorganisms in the human host. The chapter ends with several discussions on the challenges of annotating and identifying virulence factors from an automated perspective.

2.1 Microbial pathogens and virulence factors

Though recent decades have seen a decrease in mortality related to infectious disease, new dangers have appeared in the form of emerging and re-emerging pathogens in addition to the continuing threat of weaponized infectious agents [43, 57]. Underscoring the importance of this issue, the National Institute of Allergy and Infectious Disease maintains a categorical ranking of disease-causing microorganisms (NIAID Biodefense Categories) that could cause significant harm and mortality if wielded malevolently [58].

The threat of intentional agent release notwithstanding, infectious disease remains a global concern and a problem whose impact is most felt in poorer areas of the world. For example, of the more than one million deaths from malaria around the world, 85% occur in Africa; indeed the major killer of children under five years of age in Africa is malaria, just one of many tropical disease afflicting developing countries [59]. Furthermore, as urbanization expands in these countries, the risk of infectious disease incidence will likely increase (see Fig. 2.1) [7].

Fortunately, many pathogenic genomes have been sequenced, and genomic and proteomic sequences are available for many bacterial and viral causes of disease. The National Microbial Pathogen Data Resource (NMPDR), a curated database of pathogenic genomes,

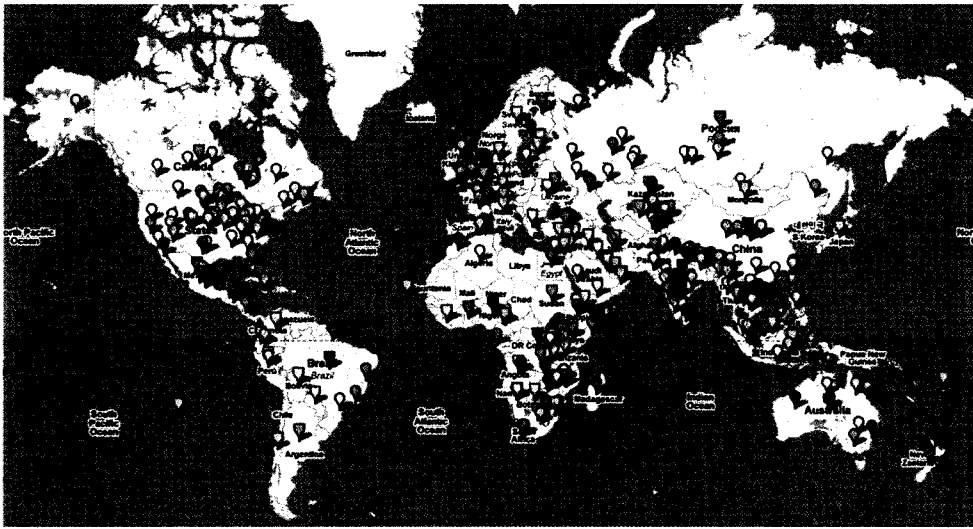


Figure 2.1: Infectious disease incidences across the globe from July 22 to August 20, 2009, colorized by urgency (deeper color is more urgent); this image was generated from [3] using data from the World Health Organization, ProMED [4, 5] and Eurosurveillance [6]. Areas of dense population, overuse of antibiotics and zoonosis tend to dominate instances of emerging infections disease [7].

lists 801 different species and strains of archaea, bacteria and eukarya infectious to mankind [26]. The availability and dissemination of this data has allowed many new discoveries in pathogenic research to stem at least partly from computational methods. For instance, Gill *et al.* [60] used sequence-based comparative genomics methods on species of staphylococcus to identify syntenic regions, and were able to draw conclusions toward the relationship transposons and insertion sequences have on the antibiotic resistance of *S. aureus*. And Setubal *et al.* developed an algorithm to predict lipoproteins in spirochetes to uncover novel genes involved in pathogenesis [61]. The challenge is thus no longer working with a poverty of data, but rather how best to mine and analyze the diverse pathogenic information available, so as to prioritize targets of study and quicken the pace of research.

2.1.1 Pathogen targets and virulence factors

One set of targets are genes within a pathogen that are directly involved in pathogenesis. In the 19th century, Robert Koch proposed a set of postulates which ostensibly defined a standard guideline for disease causation by a microorganism. Since that time, our understanding of biology and disease etiology has significantly increased and newer, gene-centric paradigms have arisen which assert that the presence of particular sequences within an organisms are a higher-resolution causative agent [62, 63]. These sequences, often termed *virulence factors*, can adopt a wide variety of functions within an organism and can have varying degrees of importance in pathogenesis.

Because the involvement of a protein in pathogenesis can be measured along a spectrum, there are several ways of determining what constitutes a virulence factor. Casadevall and Liise-Anne, and Brown *et al.* explicitly link the definition of pathogenicity to host damage, and casts it as a property that is modulated in part by both host susceptibility and resistance [64, 65]. Paine and Flower provide a broader definition: any moiety essential for causing disease in a host [63]. Wassenaar and Gaastra take a strict empirically-driven approach, and note that combined and specific evidence from phenotypic research, comparative genetics and immunological studies are necessary before a gene can be deemed as involved in virulence. They provide a hierarchy of virulence genes, ranging from those

involved in colonization and host immune evasion (“life-style genes”) to true virulence genes (refer to Tbl. 2.1) – those genes that are absent in non-pathological strains and directly responsible for damage to the host during infection [22]. For the purposes described in this dissertation, the definition used is one provided by Alksne and Projan, and modified with exceptions from Raskin *et al.* that explicitly takes into account opportunistic pathogens such as those prevalent in immune-compromised individuals: virulence factors are those genes specifically expressed during the disease process, but may not be expressed outside a human host [66, 67]. This definition would optimally exclude those genes essential to an organism for general survival outside the host, but include those which may be implicated in antibiotic resistance or *in vivo* survival.

<i>Gene class</i>	<i>Definition</i>
True virulence	Factors present only in pathogenic proteins involved in damage to host.
Colonization	Ancillary factors that aid in pathogen localization and colonization.
Defense system evasion	Factors involved in host immune evasion.
Processing virulence	Factors involved in the biosynthesis of other virulence factors.
Secretory virulence	Genes involved in secretion of virulence factors.
Virulence housekeeping	Genes that supply nutrients, or enable environmental survival within host.
Regulatory	Factors involved in the regulation of virulence factor expression.

Table 2.1: Abbreviated virulence classification proposed by Wassenaar and Gastra; the complete classification system is available from [22]. The working definition of virulence factor used in this dissertation encompasses all of the above.

While this definition provides a practical starting point when considering virulence, an important caveat is that mechanisms of virulence are often polygenic and a single factor by itself may be necessary but insufficient to cause pathogenesis. Furthermore, pathogenicity is not defined by the infectious agent in isolation – the characteristics of the host are equally, if not more, paramount considerations [64, 65]. As mentioned earlier in this chapter, an organism’s potential to affect damage to a host is a relative measure, and the interactions between the host and disease-causing organism determine the etiology, progression and

severity of an infection. At the same time, virulence factors are families of proteins whose presence in an organism is strongly indicative of an invasive tendency, such as the type III secretion system present in Gram-negative bacteria, a complex injectisome used by the bacteria to secrete effector proteins into a host cell [68]. Such virulence factors often play an important role in the generation of toxins that destroy host cells, adhesins that allow the organism to bind to host surfaces and invasin-type proteins that help them gain entry into the host cell itself [69, 70, 71].

2.1.2 Challenges to identifying virulence factors

Because mechanisms of pathogenesis can be complex, methods of classifying putative virulence proteins have arisen that reflect varying degrees of involvement in pathogenesis [70, 22]. Genes involved primarily in virulence have historically been infrequent targets for further therapeutic study, although they constitute a small subset of broad-spectrum targets [67, 72, 73]. This is true particularly in obligate or synthesis-handicapped microbes, whose metabolisms depend heavily on host processes for survival. Finding these proteins within the proteome of an entire organism can be challenging, however, because such a task requires a synthesis of skill and knowledge both about the organisms in question and the pathology and pathways involved in disease.

In the past, these difficulties have been exacerbated by the unavailability of databases devoted specifically to pathogenicity, a problem which has begun to correct itself within the scientific community (refer to Ch. 2.1.3). Nonetheless, despite the rapid sequencing of new pathogenic genomes, concrete identification of virulence factors remains difficult. Though many pathogenic genes lay within *pathogenicity islands* (PAIs), denoted by flanking repeats and GC-rich content within the genomic sequence, the exact function is still difficult to determine for any individual gene, not to mention that many virulence factors lay outside PAIs [74, 75]. Additionally, virulence factors are often a host-negative side-effect of the normal metabolic function of microorganisms within a hostile environment, such as in the case of *Y. pestis* and *E. coli*, each of which have tracts of pathogenic genes implicated in iron uptake [76, 77]. Thus, a protein, whose function may be benign in avirulent strains of an

organism may be involved with virulence in another through complex, polygenic pathways.

Concrete determination of a gene's involvement in disease is generally left to experimental results, and many studies rely on knockouts or mutations of putative pathogenic genes, often within a mouse model [78, 79]. Resulting attenuation or avirulence would then be strong evidence that the gene is involved in disease, although the exact function or role may still remain a mystery. Proper target selection is important, however, given that laboratory science makes the identification and verification of virulence factors a costly endeavor. Though these methods are often definitive, they are expensive and laborious and as such are intractable for confirming the pathogenicity of thousands of genes at a time. Thus, faster methods are preferred early on that can highlight the most likely targets before experimental assays are carried out.

2.1.3 Computational methods for finding virulence factors

Genomics-based approaches are one such group of methods, and usually revolve around variations of identifying genes that are widely conserved across pathogens, but absent in the hosts, and targeting them for further study [80, 81, 82, 83]. In the case of several pathogens, many hypothetical virulence factors have dubious homologies to genes in other organisms, making them particularly difficult to identify and characterize [84, 66, 74]. Recent studies indicate, however, that virulence factors are a fresh and attractive target for attenuating a microbe's ability to thrive and cause damage in a host [74, 85, 86, 87, 88, 89], especially as bacterial pathogens continue to evolve significant resistances to traditional antimicrobial therapies [66, 67, 72, 90, 7].

Identifying and annotating these virulence factors is an early and integral part of understanding how a disease causes damage to a host; improvements in accuracy and speed for finding proteins involved in pathogenicity have the potential to increase the analytical throughput of therapeutic research. However, prior computational research in quickly identifying virulence factors has been limited, and has only in the last few years become an area of strong interest for researchers. Several public databases have recently been released that focus exclusively on pathogenesis (see Tbl. 2.2). Among these include: the Viru-

lence Factor Database (VFDB), a repository of genomic and proteomic data for bacterial human pathogens [23, 91]; the ARGO database, a collection of virulence factors believed to be involved with resistance for β -lactam and vancomycin families of antibiotics [24]; and MVIRDB, a aggregated data warehouse of many, smaller virulence-related databases (including VFDB and ARGO) [27].

<i>Date</i>	<i>Virulence-related sequence database</i>	<i>Area of focus</i>
2005	Virulence Factor Database	General virulence factors
2005	Antibiotic Resistance Genes Online	Antibiotic resistance
2006	PHI-base	Pathogen-host interactions
2007	National Microbial Pathogen Database Resource	Annotations of pathogenic genomes
2007	MvirDB	General virulence factors
2007	Pathema	In-depth curation of selected pathogens
2008	Infectious Disease Biomarker Database	Early pathogen diagnosis
2008	varDB	Antigenic variation
2009	Antibiotic Resistance Genes Database	Antibiotic resistance

Table 2.2: Sequence databases curated specifically for virulence factors, area of specialty and relevant dates of first publication; note that this list is not meant to be comprehensive of all virulence-specific sequence databases [23, 24, 25, 26, 27, 28, 29, 30, 31].

In addition to traditional homology-based search methods, classification algorithms have been applied to the problem of virulence recognition. For example, Sachdeva *et al.* used neural networks to identify adhesins related to virulence [92]. Saha used *support vector machines* to predict general virulence factors *via* an approach similar to one proposed in [93] – mapping combinations of the amino acid alphabet to a space such that the presence of a peptide sequence would constitute a classifiable feature [94]. However, the resulting top accuracy for virulence proteins, 62.86%, was relatively low in comparison to the other protein roles predicted (*e.g.*, cellular and metabolic involvement). Work by Garg and Gupta improved on this performance by also relying on polypeptide frequencies, but in conjunction with PSI-BLAST data in a cascaded SVM-based classifier [95]. This approach yielded a much

higher accuracy of 81.8% and an area under the *receiver operating characteristic* (ROC) curve of 0.86 for generalized virulence prediction.

While initial attempts at using computational methods to identify virulence factors have met with some success, identifying and annotating genes involved in virulence remains an early and integral part of drug development. Improvements in accuracy and speed in finding proteins has the potential to increase the analytical throughput of functional determination pipelines, which is often one of the earliest rate-limiting steps in drug and vaccine development.

2.2 Protein identification and annotation

In order to prioritize genes for experimental assay, one must first be able to identify genes that have a high probability of being involved with the pathway or phenotype of interest, which subsequently involves providing a preliminary functional assignment. This process, annotation, involves assigning one or more molecular and biochemical processes to a protein, and can include additional information such as noting component domains, known sequence features of interest and the location of the protein within the cell.

2.2.1 Manual and automated methods

Manual annotation of proteins is widely recognized as time-consuming and laborious, generally requiring a scientist or team of scientists to perform numerous queries of their protein against multiple biological databases, followed by collation of the information into a cohesive story relating to its function [96, 45]. Doing so depends on many factors, including their knowledge of biology, experience with specific databases and other resources they rely on. This process is still cost-effective in comparison to traditional laboratory methods, even though it generally involves reviewing and analyzing alignments, prediction results and functional information from many sources for hundreds of records [97, 54]. Indeed, given the current clip of biological discovery, it has been posited that manual methods are incapable of keeping pace with the amount of curation necessary [98].

To help expedite annotation, many groups have attempted to address the problems of speed inherent in manual annotation by developing automated or semi-automated annota-

tion pipelines to aid in the annotation of specific species. Kasukawa *et al.* developed an annotation pipeline specific for the mouse genome, and used their systems annotations as first-pass labels for proteins that were later curated manually by scientists [99]; the ENSEMBL analysis system assigns standardized domain terms to proteins automatically, based on species-specific curated data [100].

Others have also tried to generalize the process of annotation; the FIGENIX system is a server-based annotation tool relying on multiple pipelines based on sequence, phylogeny and structure to infer functional annotation [101]; the GOblet system queries multiple, locally-stored databases to assign standardized nomenclature Gene Ontology (GO) terms to anonymous sequences [102, 103]. Accurate protein annotation often requires exhaustive searches against databases, both public and local, by the scientist(s). As it pertains to identifying which proteins are involved in pathogenicity, annotation is among the first steps. Additional approaches that have more generalizability across organisms and are more modular have been tested, with some success [19, 104].

2.2.2 Challenges to automated annotation

Despite the advantages automated annotation methods affords in effort saved, they are not without serious obstacles. Because of its speed and affordability, the primary method employed for most annotation systems is basic homology-based searching followed by automated analyses, often chained together in a *pipeline*. Unfortunately, many biological databases are rife with data of dubious quality and curation; these poor-quality annotations are often transferred from one database to another over time. Moreover, data submitted to repositories may appear correct during their time of submission given the information available, but later experiments are revealed that would alter the submission. The result of this is a significant percentage of annotations within biological databases that are either dated, poorly-annotated or even incorrect [105, 106, 107, 108]. These problems are often difficult for a skilled and knowledgeable annotator to parse through; for an automated system, it can lead to poor precision in annotation.

Triangulation across multiple and corroborating sources and methods is one way of

<i>Organism</i>	<i>Size (Mb)</i>	<i>Coding genes</i>	<i>Hypo. (%)</i>	<i>Func. (%)</i>
<i>H. sapiens</i>	3200	≤30000	50	50
<i>S. cerevisiae</i>	12.5	6000	55	45
<i>H. influenzae</i>	1.8	1750	37	63
<i>A. thaliana</i>	125	25500	48	52
<i>C. elegans</i>	97	19100	48	52
<i>L. major</i>	33	8213	64	36
<i>T. cruzi</i>	60	25041	66	33
<i>T. brucei</i>	35	10689	66	33
<i>G. lamblia</i>	12	9767	56	42
<i>C. hominis</i>	10	3994	40	60
<i>C. parvum</i>	10.4	3952	25	75
<i>P. falciparum</i>	25	5279	61	39
<i>P. yoelii</i>	25	5878	63	37

Table 2.3: The state of annotation and gene identification for the human genome, model genomes and a number of protozoan genomes of public health interest [32]. Notably, the proteomes of many infectious disease organisms remain functionally unsolved.

increasing confidence for computational predictions and increase the chances of high-quality results [54, 109, 104, 20].

Another challenge to automated annotation is related to maintenance and infrastructure; as many systems rely on different analytical and query sources, it can become difficult for such platforms to scale in phase with the number of ever-growing databases. While the *de facto* standard in almost every field for data storage, the relational database management system (RDBMS) model can be difficult to manage as the number of data sources scales up [110]. Consider that creating an RDBMS whose warehouse supports multiple databases would require a global schema which may have to be refactored when new sources are introduced; additionally, the data must be kept current, locally, across all sources.

2.3 Strengths and weaknesses of identification and annotation methods

Current published methods of virulence factor identification rely primarily on computations upon the protein sequence itself. Notably, even though there are thousands of molecular databases available, very few classification algorithms have leveraged multi-database functional information in virulence factor identification and instead rely on completely *ab initio* methods. This is particularly surprising, given that during manual protein annotation scientists regularly use multiple data repositories to triangulate a function (see previous, Ch. 2.2) and many general-purpose gene annotation systems use multiple data sources as input (*e.g.*, ENSEMBL, [100]).

It is important that any automated method identifies and annotates proteins, including those involved in virulence, first be an effective platform for *data integration* – transformation, storage and treatment of data from heterogeneous sources such that they may be queried and accessed uniformly. As emphasized by Wong [111], any method that integrates heterogeneous data for bioinformatics tasks must meet several criteria, among which include independence from external source schemata, a robust internal data model and loose couplings between the internal model and queries to reduce maintenance costs associated with external change.

Identification and classification of virulence methods is further hampered by the limited holistic understanding virulence, the lack of a centralized vocabulary for describing a protein's involvement in pathogenicity and the only very recent arrival of multiple sequence databases focused on virulence genes. Furthermore, much of the data that is available in virulence and non-virulence databases can be specious in nature without human input and curation.

Chapter 3

METHODS OF DATA INTEGRATION AND STATISTICAL CLASSIFICATION IN BIOLOGY

The previous chapter (Ch. 2) outlined the challenges involved in annotating and identifying virulence proteins, and asserted that there were serious shortcomings of traditional approaches to gene annotation as it relates to predicting virulence. Some of these obstacles were endemic to annotation in general, while others were specific to the area of virulence factor identification. The following chapter discusses two separate computational methods in a general sense – machine learning and data integration – and provides the supporting groundwork for the use of a combination of these two approaches to address problems in identifying and annotating proteins involved in pathogenicity.

3.1 Computationally predicting protein function

One family of methods that have been used extensively in the past for protein annotation, and thus holds promise for virulence factor identification, is *statistical learning* or *pattern recognition* algorithms. This is a broad category that encompasses a variety of computational approaches capable of either classifying instances into any number of categories, or, as is the case in regression, relate characteristics of instances to some predicted real value. The following section will review several popular methods and specifically focus on *supervised* methods – algorithms and approaches that rely on known prior instances to classify unknown future instances – which have been used with success with protein function prediction and computational annotation in the past.

3.1.1 Supervised learning

Formally, the problem of supervised class learning can be expressed simply as finding some function which defines a mapping or probability from any given instance or example to

some class or label. Each instance is represented by a set of *features*, which ostensibly are of some predictive value to discerning the correct label for the instance, either individually or in combination. For example, a simple yet effective feature for a protein sequence learner on cellular localization could be a boolean representing the presence of a signal peptide, with 0 for the absence of a signal peptide, 1 otherwise.¹

The problem of learning can then be further elaborated, where given a set S of m instances represented by n features with known corresponding labels taken from \mathcal{L} :

$$S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m) | y_i \in \mathcal{L}, \mathbf{x}_i \in \mathbb{R}^n\}, \quad (3.1)$$

define a function f and parameters θ such that for any given instance \mathbf{z} of unknown label:

$$f(\mathbf{z} | \theta, S) = y. \quad (3.2)$$

The above is a basic model for classification that applies to a large number of learning methods used in biology, some of which are outlined below.

3.1.2 Review of learning methods in biology

Approaches based on probabilistic belief networks have been popular in the past for cases where information can be expressed in a conditionally independent manner. Networks such as those grounded on Bayes' theorem provide a graphical way of representing data when their relationships are suspected to be causal in nature. In cases where the networks represent biological interaction, the function prediction is done *via* 'guilt by association'. For example, King *et al.* [112] used this approach to predict gene function using patterns of Gene Ontology (GO) [102] annotation, under the hypothesis that the co-occurrence of terms for *S. cerevisiae* can be used to identify as-yet undiscovered functions. Comparing Bayesian network and decision tree approaches with a model where annotations were assumed independent, King *et al.* found that both approaches performed well and that while a Bayesian network yielded a higher *area under the ROC curve* (AUC), decision trees did better at lower false positive

¹Notably, this particular feature could be made richer by expressing the presence of a signal peptide by its location within the sequence.

rates. King *et al.* further applied the decision tree model to prediction of gene and phenotype relations, with similar albeit less optimum results [113].

While the work by King *et al.* relied primarily on a single data source, prior research by Troyanskaya *et al.* integrated multiple data sources over Bayesian methods to predict paired protein function [114]. The system developed, called MAGIC, uses protein-protein interaction, binding site and clustered microarray expression data to gauge the probability of the products of two genes interacting. Functional gene groupings are generated as a result, and can be used to infer the function of some protein based on other, known interacting proteins.

The combination of heterogeneous information for use in protein function prediction has formed the basis for other approaches as well. Deng *et al.* used a *Markov random field* (MRF) to predict protein function using a variety of sources [115]. They compared this integrated approach with a baseline method that used only a single data source (physical interaction data), and found that the use of multiple data sources resulted in a considerable 30% increase of precision. And Chua *et al.* [18] used graph-based methods to generate a weighted graph of estimated confidences built from different sources; a layered, simple voting scheme was then employed to determine function. This approach, termed *Integrated Weighted Averaging* (IWA) is computationally inexpensive, and compared favorably with methods that are more machine-intensive. In fact, the AUC of the IWA method improved when the input data was made up-to-date, underlining the transitory nature of biological information as it pertains to functional annotation.

Another family of methods capable of discrimination and pattern recognition does so by forming a dividing plane between classes. Using the notation in (3.1), these methods operate by envisioning vector instances from S as points within an n -dimensional space, whereupon a *separating hyperplane* is found that forms the *decision boundary* between classes in the space. For a formal description, define a binary classification problem with $\mathcal{L} = \{1, -1\}$ and instances S . Identify a vector \mathbf{w} normal to some hyperplane in \mathbb{R}^n and offset b such that:

$$1 = y_i(\mathbf{w}^T \mathbf{x}_i + b), \text{ for } i = 1, \dots, m. \quad (3.3)$$

Finding the separating hyperplane for \mathbf{x} is thus found by solving the constrained optimization problem (in *primal form*):

$$\min_{\mathbf{w}, b} \frac{1}{\|\mathbf{w}\|^2} \quad (3.4)$$

subject to $1 \leq y_i(\mathbf{w}^T \mathbf{x}_i + b)$, for $i = 1, \dots, m$,

which can be expressed in terms of instances rather than features *via* the *dual form* [116, 117]:

$$\max_{\alpha} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to

$$\alpha_i \geq 0 \text{ for } i = 1, \dots, m,$$

$$\sum_{i=1}^m \alpha_i y_i = 0. \quad (3.5)$$

Notably, the above assumes a hyperplane exists that can perfectly separate the two classes, and thus a separating hyperplane may not be found if the classes are nonlinear in \mathbb{R}^n . To allow for a *nonlinear* mapping, some transformation mapping Φ may be applied to \mathbf{x} , where $\mathbf{x}_i^T \mathbf{x}_j \Rightarrow \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \Rightarrow k(\mathbf{x}_i, \mathbf{x}_j)$; this defines the *kernel function* and allows the features of \mathbf{x} to be mapped to some space where a linear separation *may* exist (refer to Fig. 3.1).

Adapting (3.5) with a kernel function, and setting a minimum *margin* C from the separating hyperplane yields [117]:

$$\max_{\alpha} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to

$$0 \leq \alpha_i \leq C \text{ for } i = 1, \dots, m,$$

$$\sum_{i=1}^m \alpha_i y_i = 0, \quad (3.6)$$

which is similar to the form given in (3.5), save the constraints on α and the primal minimization, $\frac{1}{\|\mathbf{w}\|^2} + C \sum_{i=1}^m \xi_i$, where ξ is a slack variable allowing for some classification

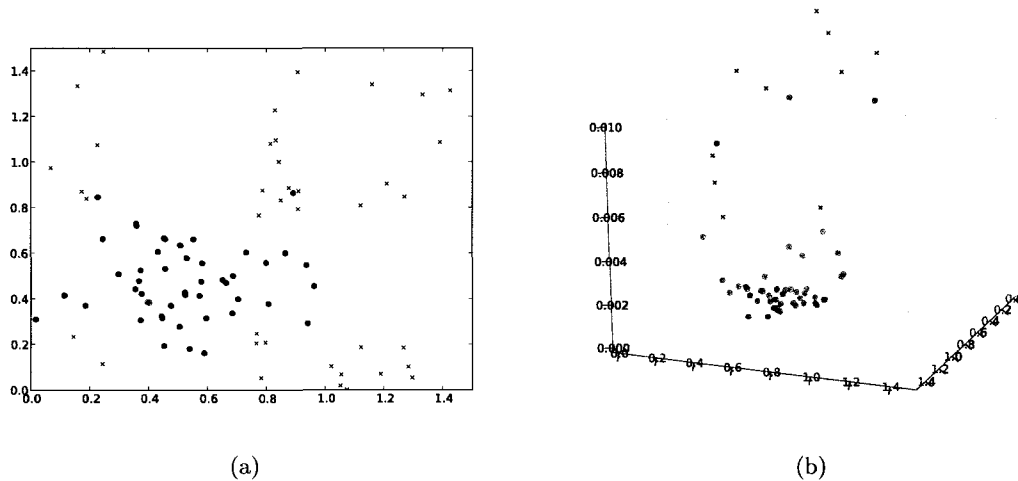


Figure 3.1: Above, Fig. 3.1a shows a series of positive (red circles) and negative (black crosses) instances which are not linearly separable without large error in their native 2D space; Fig. 3.1b shows the same data plotted with a mapping (Φ) to a 3D space (with $z = (xy - \frac{x}{2} - \frac{y}{2} + \frac{1}{4})^2$) where a linear separation is possible with less error.

error up to a point such that $\xi_i > 1$ indicates a misclassification for \mathbf{x}_i thus forming a *soft margin*. The above defines a *support vector machine* (SVM) [116], and is generally solved using convex optimization methods [118].

Capable of both regression and discriminative classification, SVMs offer several advantages over other learning algorithms. They are fairly resistant to overfitting, as the optimal hyperplane is chosen that maximizes the margin. This is in comparison to other learning methods, such as *neural networks* where a solution may not be optimal despite correct classification of the training set. Additionally, using a soft margin makes the training process for an SVM less sensitive to noise. Because of these appealing characteristics, as well as accuracy for nonlinear tasks, SVMs have been used for analysis and classification of biomedical data with resulting performance comparable to or better than other methods in many cases. Brown *et al.* apply SVMs with various kernels to the task of classifying genes into five Munich Information Center for Protein Sequences (MIPS) functional categories, compared their

performance to other non-SVM kernel methods and decision trees, and found that SVMs outperformed these other methods [119]. SVMs have also been used for secondary structure prediction, and in work by Hua and Sun it outperformed other approaches, including one based on consensus of multiple methods ([120]). Methods for classification based on protein sequence data have also been used successfully. Leslie *et al.* used a *spectrum kernel* whose input are the frequencies of protein words of varying length, and show that the accuracy at classifying proteins by SCOP (Structural Classification of Protein) is comparable to that of other, more computationally-expensive methods [93]. And, when this model is adapted to allow for gaps, it actually outperforms other state-of-the-art methods [121].

An additional advantage is that SVMs may be used to combine multiple sources of data in a variety of ways. One such method is the direct integration of the SVM's vectors, or *early integration* [122], where the feature vectors of two or more sources are concatenated, forming a vector for each instance whose length is equal to the total number of features across all sources. In *intermediate integration*, the kernels (K) themselves are added, *i.e.*, [123]:

$$K' = \sum_i \mu_i K_i, \quad (3.7)$$

where μ_i is a weight for each source kernel K_i representative of the importance or affect each data source should have on the data; optimal weights can be determined *via* a grid search or *quadratically constrained quadratic programming* [124, 125]. The SVM is then trained on the aggregate kernel, K' . In late integration, source kernels are classified individually, and the distances from the respective decision boundaries are summed for the final result. These methods have been used for protein classification, DNA/RNA-binding protein prediction and prediction of metastatic cancer from clinical and microarray data [122, 126, 127]. Finally, in a more sophisticated form of late integration, SVMs can be used in such a way as to form the inputs to yet another learning method. Obozinski *et al.* used this approach for a regressor of consistent probabilistic values for GO terms [128], while Garg and Gupta used a *cascaded SVM* approach to predict protein-relatedness to virulence based on spectrum and PSI-BLAST kernels [95], the problem of interest for this dissertation.

The latter methods involved the use of multiple data sources of data in performing classification tasks. Scientists now look to examining as many views on their proteins of interest in order to draw conclusions; relying on many different forms of information is one way improving computational techniques by exploiting complementary and supporting information derived from them. Notably, many of these techniques often include experimental data sources, which are generally considered to be of the most trustable quality and highest provenance; in combination with less expensive methods, using many sources of data can produce impressive results.

3.2 Data integration methods in biology

Generally, experimentally verified information will be difficult to obtain, particularly for newly-sequenced genomes or in instances where proteins are difficult to work with in a laboratory. In these cases, relying on whatever sparse and speculative information is available from public biological databases *via* sequence alignments, statistical analyses, structure prediction, *etc.* is often the only place to turn for possibly pertinent information. Use of these resources, however, comes with its own challenges apart from those encountered in the laboratory.

To any individual scientist, biological data relevant to their research is scattered across numerous and fractured databases, many of which have their own idiosyncratic interface, usage and data model. As a result, processes and analytical protocols that involve many databases can be difficult and tedious, and users of the databases must learn each individual source in order to take advantage of the complementary information available in repositories that contain information on different domains [129]. This problem is exacerbated by both the continued growth of the sheer number of biological databases and the dynamic nature of their content.

There are a variety of methods that have been in use to address this problem, most of which rely on some variation of centralized data integration that allows multiple data sources to be queried *via* a single interface. Primarily, these approaches fall under two categories: *data warehouses* and *federated systems* [110]. In a data warehouse remote information is pulled, cleansed and then stored locally, generally within a traditional RDBMS; the local

copy of the data stored under this architecture is transformed physically from one data model to another. Benefits of this approach include query speed and greater control over the quality of data, and are juxtaposed against the disadvantages of limited guarantees for data currency and practical limitations on scalability. Previous applications of data warehouses to the domain of biology include ENSEMBL [100], ATLAS [33] and GUS [38], all of which have been used for data curation and annotation. For researchers dealing with multitudinous data and complex, expensive queries, the data warehouse approach provides a strong advantage in time to query retrieval and content control.

To illustrate the disadvantages of an RDBMS warehouse, consider that as individual data sources evolve the data model within an RDBMS must evolve to match as well, requiring database refactoring and possibly renormalization. These changes, in turn, may render previously valid queries obsolete. Given that biological databases change frequently and the increasing scale of repositories necessitate regular changes to the RDBMS schema, maintenance on data warehouses for biology can be very challenging.

Under a federated architecture, data remains at the source, subsequently absolving the need for a locally-maintained RDBMS. Instead, a tiered architecture is used that allows reformulation of user queries to each of the individual sources for data retrieval. A major shortcoming of this approach is the time it may take to resolve a query. However, because the data remains at the source and the job of maintenance is left to the originators, this approaches scales particularly well for *ad hoc* treatment of data sources relative to a warehouse-based system. Additionally, data retrieved under such a scheme is clearly guaranteed to be as current as the data within the source itself.

The plot in Fig. 3.2 is a strong motivation to rely on federated architectures for comprehensive data integration. As the number of databases increases (and thus the number of potentially relevant repositories for any one domain of research) the cost of maintaining a federated system for uniform access will linearly increase; evolving a data warehouse to keep pace at the same level of cost would be more difficult. The federated data model has been employed successfully in the SRS[40], K2/Klesli [38] and TAMBIS [41] systems, among others.

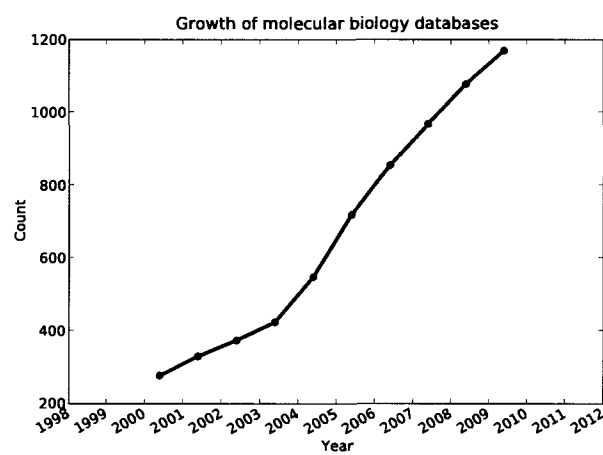


Figure 3.2: The linearly-increasing number of biological databases from 2000 to 2009 [8, 9, 10, 11, 12, 13, 14, 15, 16]; it would be physically near-impossible for a single scientist to manually parse through nearly up to 1200 databases in search of relevant information in a timely fashion.

3.2.1 Federated data integration

For the reasons outlined in the previous section, the work of this dissertation relies on the federated architecture, and one driven by a specific type of schema. Following the syntax put forth by Lenzerini in [130], we can formally define an architecture-independent data integration system \mathcal{I} as $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where \mathcal{G} is the *global schema*, \mathcal{S} the *source schema* and \mathcal{M} the set of *mappings* between \mathcal{G} and \mathcal{S} . In the most conservative sense this formalism of data integration is reliant on source and schema explication, and thus the mappings $m \in \mathcal{M}$ that form the correspondences between \mathcal{S} and \mathcal{G} ($m_{\mathcal{G}} \mapsto m_{\mathcal{S}}$ and $m_{\mathcal{S}} \mapsto m_{\mathcal{G}}$) are specific for all data sources \mathcal{D} described by \mathcal{S} . In other words, formulating a query q over \mathcal{G} whose results would be satisfied by multiple data sources would require individual enumeration of those sources in q .

To illustrate support for more expressive queries, we further define the components as follows: let \mathcal{G} be constructed from some collection $\Lambda_{\mathcal{G}} = \{g_1, g_2, \dots, g_i\}$, where g represents a *mediated* class or entity, and \mathcal{S} constructed from some collection $\Lambda_{\mathcal{S}} = \{s_1, s_2, \dots, s_j\}$ where s is a valid element from any data source $D \in \mathcal{D}$. Mappings from \mathcal{G} to \mathcal{S} are handled as $m(g) \mapsto m(s)$; then let:

$$\forall g : \quad g \subseteq g \quad (3.8)$$

$$\forall g_i, g_j : \quad g_i \subseteq g_j \wedge g_j \subseteq g_i, \text{ then } g_i = g_j \quad (3.9)$$

$$\forall g_i, g_j, g_k : \quad g_i \subseteq g_j \wedge g_j \subseteq g_k, \text{ then } g_i \subseteq g_k \quad (3.10)$$

which imposes a subsumptive ordering over the schema entities of \mathcal{G} which enables more expressive queries. The above relations can be likened to a frame-based hierarchy, whereupon if the hierarchy is restricted to singular inheritance:

$$\text{if } a \subseteq b \text{ then } q_a(c) \subseteq q_b(c), \text{ where } a, b \in \Lambda_{\mathcal{G}} \quad (3.11)$$

holds, where q_v is some query with variables c of arbitrary arity. This hierarchically-driven (per the ordering on \mathcal{G}) construct will be referred from now on as a *mediator-based*, or a

mediated schema, and allows queries on elements of \mathcal{G} that support *inheritance* and implicit *containment* (as in (3.11)) not possible under a normal global schema. Depending on the mappings, then, a single query may be posed on \mathcal{G} without individual enumeration of sources. Furthermore, other relations on the elements of \mathcal{G} beyond queries may have similar properties as (3.11).

Example We define a mediated schema with the following elements and properties:

$$\begin{aligned}\Lambda_{\mathcal{G}} &\leftarrow \{Sequence, DNASequence, ProteinSequence\} \\ DNASequence &\subset Sequence \\ ProteinSequence &\subset Sequence\end{aligned}$$

and queries:

$$\begin{aligned}R_1 &\leftarrow q_{Sequence}(c) \\ R_2 &\leftarrow q_{ProteinSequence}(c) \\ R_3 &\leftarrow q_{DNASequence}(c)\end{aligned}$$

where R_n denote query results. Then the following holds true:

$$\begin{aligned}R_3 &\subseteq R_1 \\ R_2 &\subseteq R_1.\end{aligned}$$

From an intuitive biological perspective, the above states that given the same schema and sources across queries, the results of specific queries on protein and DNA sequences will be contained by a broader query on general biological sequences. \square

The federated, mediated approach described allows queries to be posed completely on a single, unifying schema in terms of $\Lambda_{\mathcal{G}}$ without regard to the individual sources. Ideally, the schema would be broadly representative of some domain of interest, such as functional annotation of proteins, yet specific enough to have utility in querying and retrieval.

Federated data integration systems have been employed in biology in the past, both informally and formally. On the informal side, the ENSEMBL system is a *pipeline* for sequence analysis and automated annotation [100]. ENSEMBL is composed of many, local analytical components specifically chained together in such a way as to offer a comprehensive analytical platform for sequences of interest. A management module (the ‘Rule Manager’) controls invocation of the components, and new components may be programmatically composed and linked to the appropriate section of the pipeline. The management module is not a formal generalized data integration system, however, and the schema is not mediated. Furthermore, the ability to query is extremely limited, as system interaction is strictly defined by the structure of the pipeline. And, any new components are specifically designed to fit in particular sections of the pipeline. As such the system is not scalable in the face of a large number of analyses.

At the middle end of the spectrum, semi-formal integration systems include the *Sequence Retrieval System*, or SRS, which provides a federated architecture to accessing multiple sources described *via* linked flat-files [40]. While the SRS model is more generalized and does provide uniform access to the over 400 sources it supports, the prime use of SRS is as an indexing and search tool. Sources are parsed for text and field names and indexed in a relational database, which in turn is used to manage query retrieval. Because of this, SRS is sometimes characterized more as a user interface integration tool, and less a formal and fully-expressive data integration system [131].

A federated data integration systems with a centralized mediated schema is TAMBIS, which stands for *Transparent Access to Multiple Bioinformatics Information Sources* [41]. Unlike previously described systems, TAMBIS supports expressive user-posed queries *via* the CPL (Collection Programming Language) query language. Furthermore, TAMBIS is driven by a common data model that, though limited, describes the domain of interest and allows queries to be made independent of individual data sources. Whereas in the case of ENSEMBL, SRS and systems like them that require a non-trivial level of user knowledge of the sources for understanding, interaction with TAMBIS is in the language of the central schema. TAMBIS thus fits the model of a true federated data integration system, supporting both a centralized schema and a mechanism upon which expressive queries may be made

using that schema.

On the other hand, effective interrogation of TAMBIS is predicated on the user’s knowledge and ability to form specific and directed queries in CPL. Thus, like the other systems described, TAMBIS is limited by the technical expertise of its users and serves as a barrier to usage. Also, while there are many cases where a biologist may be interested in posing a specific query, there are other cases, particularly in early research, where one may be interested in generally acquiring as much information as possible regarding a gene or protein. The following section discusses a querying approach for integration systems that extends the federated model graphically. This path-based view of integration is effective at quickly discovering more information and provides a querying mechanism that is arguably easier than using directed queries.

3.2.2 Path-based federation

An extension of the method outlined in the previous section (Ch. 3.2.1) that fits particularly well in the paradigm of biological databases is managing retrieved data in a *path-based* manner [132, 133, 36]. This model begins with an initial query, and through a series of links made explicit in the schema, joins are made across databases connecting information many hops removed from the original query. The result is a *query graph*, which can be rapidly grown *via* continued re-querying of acquired data. If this is done in an exploratory fashion, the query graph will contain information of varying relevance to the original query; this action is termed an *exploratory query* [20].

Formally, the query graph $G = \langle V, E \rangle$ is composed of nodes V , representing queries or results, and edges $E \subseteq \{V \times V\}$ which are the relations between the nodes. Joins are accomplished using references of non-materialized nodes from materialized nodes, and are very much characterized similarly to the relational join concept; materialization of the edge between v_i and v_j is $v_i \bowtie_d v_j$, where $v_i, v_j \in V$ and d is some attribute (naturally occurring or artificially generated) in v_j with primary key-like qualities. Nodes are populated according to the elements of Λ_S , but map to elements of Λ_G ; these correspondences are concrete instances of the mappings in \mathcal{M} , $m_G \mapsto m_S$. Thus, the query graph is essentially a

materialization of the mediated schema with instances of entities from data sources in the form of nodes.

Example We define the following query and databases (with elements expressed in terms of Λ_G) for some biological data integration system:

$$\begin{aligned} q &\leftarrow gene(a) \\ D_1 &\equiv (gene(a), protein(b), e(a, b)) \\ D_2 &\equiv (protein(b), family(c), e(b, c)) \\ D_3 &\equiv (family(c)) \end{aligned}$$

which yields the following result on execution of q on D_1 following path-based query expansion:

$$R_q \leftarrow gene(a), protein(b), family(c), e(a, b), e(b, c).$$

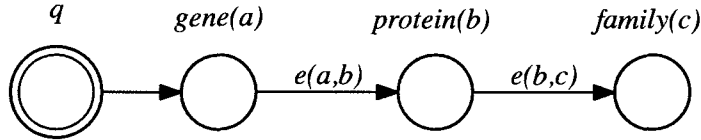


Figure 3.3: Graphical representation of the query expansion for $q \leftarrow gene(a)$.

In the above, $gene$ is related to a $family$ through their respective databases *via* the join $D_1 \bowtie_{protein} D_2$ (see Fig. 3.3). \square

This formalism of representation and querying allows integrated data sources to be *browsed* in addition to being *queried*. In this model, users do not need to necessarily have

in mind what their particular interest is – they may now merely follow cross-references from source to source. While this approach may seem more tedious, and indeed may be in cases where a biologist is interested in determining a specific answer to a question, it is a more accurate reflection of how some biological tasks are carried out manually, including protein annotation. Though simple, it is a powerful and expressive model of data integration when applied to biological databases, many of which cover a variety of domains but often retain cross-references to other related databases [134].

Later federated data integration systems apply this model of data retrieval. Some systems even extend previous models to support path-based federation, such as BioGuideSRS [34], which provides a graphical path-based interface to SRS. Users may browse the sources in SRS using point-and-click interactions, and define preferences, such as the minimum number of cross-references per path. BioGuideSRS is thus an example of a system that uses a comprehensive, triangulation-based approach to presenting and querying information; through path-based browsing, a scientist can visibly see *via* sources and paths how a result is reached.

Another example of such a system is BioMediator, a generalized data integration system that has gone through several iterations, the latest of which supports graph-based browsing [132, 135]. Data retrieved by BioMediator is described using a richly-annotated mediated schema, with concepts featuring the characteristics as described in (3.8-3.10); indeed, even edges in BioMediator are organized within a hierarchical model. Similar to BioGuideSRS, users may make queries to a graph-based interface and browse by expanding and following links. Built on a tiered and decoupled architecture, BioMediator has been in the past adapted to several biological tasks, including expression array analysis [136], gene annotation [19, 104] and processing of neural information [137].

3.3 Discussion

In this chapter, two fields important to information mining and management for biomedicine have been briefly reviewed: machine learning and data integration methods. The former has been used frequently in biological research as a way of labeling and classification, while the latter as means of querying and gathering large quantities of information. Moreover,

current research has trended toward a combination of these approaches – utilizing learning methods on combinations of data. These approaches, however, have been *ad hoc* and often any integration is done only at the *data-level*; no previous research has attempted *query-level* integration with statistical learning techniques.

The remainder of this dissertation discusses the methods and results for using a data integration systems as a means of aggregating and mapping stored biological data for virulence role prediction. Immediately following, Ch. 4 outlines the implementation of a novel data integration system that builds off the successes, and improves on the methods, of previous data integration platforms. This system is specifically designed to accommodate high-throughput, repeatable biological queries, the type of which current integration systems are unsuitable to address for a number of reasons. For example, while systems such as BioGuideSRS and BioMediator provide scalable approaches to both integrating and querying data, they have been designed primarily to serve in an ‘active session’ role. That is, the systems’ model of usage are user-centric, and designed such that a biologist would query and search in an interactive manner. In order to support high-throughput proteomic analysis of the type needed for learning query-level data, a framework for data integration is needed that supports autonomous querying and data processing.

Chapter 4

**MODEL AND IMPLEMENTATION OF A BIOLOGIC DATA
INTEGRATION ENGINE**

The previous chapters have outlined challenges to general function prediction as well as methods in both the fields of data integration and machine learning used to address these challenges. A specific subproblem of function prediction, virulence identification, has also been discussed as well as recent advances in the emerging field of automation and computation for pathogenic proteins. This chapter¹ goes beyond this background by proposing a model and implementation of a federated data integration system for biology. The design of the system proposed was motivated by addressing the challenges of scalable data integration faced in biology, with the added flexibility of rapidly accommodating new features, including those to support statistical learning for virulence identification.

Relying on path-based exploratory queries, this system can pose singular queries uniformly across data sources, and return results recursively according to a mediated schema. Exploratory queries are effective at generating large quantities of information, albeit of varying relevance. Querying this way does not require the user to learn a possibly complicated query language, and when coupled with a graphical user interface can be effective in conveying dominant information regarding the original query within the space of biological databases.

Among prior research that deals with mining data from integrated biological sources, the work by Cohen-Boulakia, *et al.* with BioGuide, Lacroix *et al.* with BioNavigation, and BioMediator, stand out as the most comparable in terms of the data integration approach taken [132, 36, 138] (also refer to Tbl. 4.1). These systems rely on path-based querying over a graphical representation of the results, accommodate a browsing mode for exploratory

¹The methods described herein for Ch. 4.1.4 were presented at the American Medical Informatics Association 2009 Fall Symposium as “Supporting retrieval of diverse biomedical data using evidence-aware queries”; a paper of the same name by Cadag and Tarczy-Hornoch is under consideration for journal publication as of Nov. 1, 2009.

queries and use explicit mappings of predefined entities to multiple sources for increased flexibility, all of which is used in the implementation herein.

4.1 System architecture

To address the shortcomings of current federated integration systems, a system was developed for the explicit purpose of supporting integration over heterogeneous biological data sources while at the same time affording the architectural flexibility and scalability to support high-throughput queries *via* a number of mediums. The purpose of the latter point being that such a system would make data retrieval and processing more amenable to statistical learning over the results. The *MIQAPS* framework (for “**M**ultisource **I**ntegrated **Q**ueries **A**gainst **P**rotein **S**equences”, pronounced *mai-kaps*) was developed as a query, integration and retrieval engine for exploratory and structured queries. The primary components of the data integration system developed are similar to those of a traditional federated data integration systems [139, 140, 132, 130]: a mediated schema, a data directory and source catalogue, interfaces to various data sources and a browsing engine. Using *MIQAPS*, it is possible to query using a protein sequence and retrieve similarity-based results across different heterogeneous resources, a practice that is amongst the most basic of needs in modern molecular biology laboratories. System implementation was done in the Python programming language [141].

Driven at its core by a general-purpose data integration engine², *MIQAPS* is similar to several other data integration systems (see Tbl. 4.1). The architecture is close to that of *BioMediator*'s, with some notable operational differences, the largest of which are that it was designed to be extremely lightweight and low-overhead, and allows caching within a relational database, *via* PostgreSQL³. Like *BioMediator*, however, *MIQAPS* supports exploratory queries of the kind useful for retrieving a large amount of information regarding a query. As a result, queries can be posed in an unguided way (as described below), which may be the most convenient for biologists uninterested in learning a path-based

²PyDI, <http://pydi.sourceforge.net>

³PostgreSQL, <http://www.postgresql.org>

query language. Yet at the same time, the potential exists for expressive and specific query capabilities using a structured query language in the form of DaRQL [142]. Its scaled, extensible architecture allows for easy coupling with other systems or components, and for this research MIQAPS was joined with a graph analysis engine⁴ so that results from queries in MIQAPS are automatically mapped onto a manipulable graph for examination.

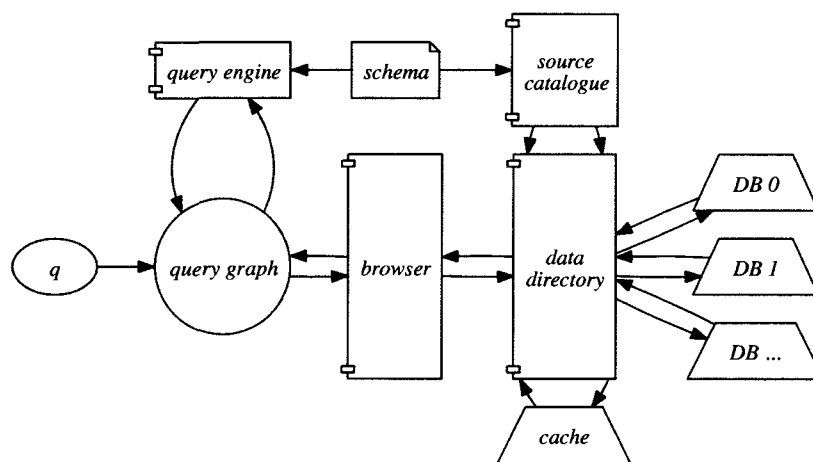


Figure 4.1: Architecture of the core MIQAPS system, where arrows denote information flow between the various components (dotted indicates a non-required interaction). Queries originate on the left (q) and are posed to a *query graph* instance either in an exploratory (key-value) fashion or using a path-based structured query language.

Exploratory queries against MIQAPS are done in an entity-attribute-value tuple against the entities within the *schema*. For example, a protein sequence query would be posed *via* the tuple: (*Seed*, *ProteinSequenceQuery*, #*ID*, < *seq* >), where *Seed* is the local source, *ProteinSequenceQuery* an entity in the schema, #*ID* a reserved term representing the unique identifier for all entities and < *seq* > is the query sequence. At a high level, an exploratory query is executed in the following manner: a client seeds a sequence query to the

⁴NetworkX, <http://networkx.lanl.gov>

browser in the form of an entity-attribute-value tuple, valid types of which are enumerated within the schema. The browser communicates with the *data directory*, which performs a look-up within the *source catalogue* of the data sources to determine valid sources. Information regarding queryable entities is stored in the user-defined schema, which explicitly defines the valid mappings between data sources as well as the structure and hierarchy of the entities used as mediating concepts across sources. The data directory manages the individual interfaces to each data source (denoted by *DB x* in Fig. 4.1), which are modularized and separate from the core engine. Data that is returned may themselves be queryable, and thus continued querying of results amount to sequential joins across numerous databases (refer to Ch. 3.2.2). Once translated by the data directory into materialized schema entities, attributes and relations, results are sent back to the browser in the form of a *query graph*, where nodes represent individual data records mapped to entities in the schema, and edges the cross-referencing relations between those entities.

4.1.1 *Frame-based mediated schema*

The core of the constituent data integration engine, and thus of MIQAPS, is a user-definable mediated schema. This schema manages the translation mapping of queries posed to MIQAPS to the native query format of each individual source, as well as the translation mappings from the sources into entities. All of these components are stored within a single flat-file read during start-up, and are conceptually represented *via S-expressions*.

Entity definitions conceptually form the mediated elements of the schema; records from each data source that are valid within MIQAPS map to some entity. The entities are inter-related through a frame-based hierarchical system, as formally described in Ch. 3.2.1. This allows support for inheritance across concepts represented (per Eqs. 3.8, 3.9 and 3.10), and is composed of three primary components: the *entity definitions*, the *source definitions* and the *link declarations*. A natural consequence of this is that provided the schema is well-constructed and is a fair reflection of the domain and information of interest, queries posed to a concept in the schema (and any associated data sources) allows axiomatic duplication of that query to the appropriate sub-concepts (and any further associated data sources). The

```

(:cls < entity_name >
  ( {(:isa < parent_name >)} )
  ( {(:atr < attribute_name > .|*)} )
)

```

Figure 4.2: Formal entity declaration in the MIQAPS mediated schema in extended Backus-Naur form. The ‘.’ symbol denotes an attribute of single arity, while ‘*’ indicates an attribute of n -nary arity.

entities also form the language, which define the valid cross-reference mappings between data sources, in tandem with explicit sources. Fig. 4.2 displays the format of valid schema entities in the S-expression representation. A valid example of this is the eponymously-named top-level entity whose definition is: ‘(:cls Entity () ())’; it is a parent of the query entity: ‘(:cls Query ((:isa Entity)) (:atr QueryString .))’.

Source definitions (shown in Fig. 4.3) specify which data sources are valid under the schema, and define the mappings between schema entities and records returned from each source. The source definitions contain extensive information on source-specific querying, mapping and importantly instructions on *link reification*, which allows the formation of an edge between two entities in the query graph. Each source is composed of any number of entities (defined *via* the entity hierarchy), and for each of these entities valid edges to the entity are defined (in the form of the :qry tuple). The :map tuple indicates how individual information returned from a source are translated into attributes of the entity. In some cases, it is useful to map to attributes that are not defined in the corresponding entity, and if so the :prp tuple is used to form a bindable symbol on-the-fly. These symbols allow for the instantiation of source-to-source links without having to specify this information in each entity explicitly. Because some information collected real-time may be useful for filtering purposes (*e.g.*, using alignment scores as an annotation between two sequences) and to prevent needless querying, parameters (:par) may be added to a source-entity combination. These parameters allow the schema to be tweaked for filtering without requiring code changes to the source interfaces. For example, one possible parameter might be (:par


```

(:src < source_name >
  ( {(:cls < entity_name >
      (:trg < label >
        ( {(:prp < reified_attr > .|*}) )
        ( {(:map < src_attribute > < attribute >)} )
        ( {(:qry < entity_name >
            ( {(:prp < attribute > .|*}) )
            ( {(:par < name > < value >)} ) ) } ) } )
  )

```

Figure 4.3: Formal source definition in the MIQAPS mediated schema in extended Backus-Naur form.

E-value-min 1e-35); this information could be used by the source interfaces as a minimal condition to conduct a recursive query.

Finally, link declarations are the abstract representation of the edges between nodes in the query graph (formally described in Fig. 4.4). Each link is composed of two source references: one for the originating source (the *tail*) and one for the arriving source (the *head*); this permits the same source to be referenced in both cases, if there are entities internal to the source that are linked. The generator tuple (:gen) specifies the interface which is queried in order to materialize the head node in the link. The mapping between this query and the concept in each source is *via* the :qry tuple, present in both the link and the source definitions. However, it is at the link level where the edge attribute mappings are applied in the :map tuple, which maps information labeled from the data source as a reified attribute on the edge.⁵

Management of the three components of the schema fall on the source catalogue and data directory, the former handling the loading and instantiation of the objects into the hierarchical representations, source definitions and linkages, and the latter whose concerns are translation of returned results into schema form and data storage. While requiring

⁵For concrete examples of how entities, sources and links are represented within MIQAPS, refer to App. A.1.

```

(:lnk < link_name >
  (:gen < generator_name >)
  (:src < source > (:cls < entity > (:prp < attribute >)))
  (:src < source > (:cls < entity > (:qry < query_name >)))
  ( {(:map < label > < attribute >} ) ) )
)

```

Figure 4.4: Formal link declaration in the MIQAPS mediated schema in extended Backus-Naur form.

considerable initial overhead, this layered system of mappings cleanly separates the retrieval of data on the source side from the management of the data once it has arrived in MIQAPS. As commonly happens in biomedical resources, access and format of the data evolves over time. Management of these changes within an integration system can be difficult and arduous, such as when the alterations affect cross-linkages and references between databases. It becomes even more difficult if queries with complexity beyond exploratory are necessary – entities and mappings must be retained in order to maintain the validity of past queries. This architecture addresses these challenges with clear divisions between a source layer (the interfaces), a query transformation layer (schema, data directory and source catalogue) and an interaction layer (browsing engine, query engine, query graph, discussed below); each of these layers may be changed with little impact to the others. Fig. 4.5 shows a schema used in MIQAPS, which relies on several different data sources of varying types.⁶

4.1.2 Query graph and browsing engine

The query graph and browsing engine are the primary mechanism through which a user interacts with the data integration system. In the case of exploratory queries made on MIQAPS, it is the browser which performs the planning involved in determining which

⁶Fig. 4.5 appears in a prior publication by Cadag, Tarczy-Hornoch and Myler entitled “On the reachability of trustworthy information from integrated exploratory biological queries”, in *Lecture Notes in Computer Science* (Data Integration in the Life Sciences 2009) 5647 pp. 55-70 with kind permission of Springer Science+Business Media.

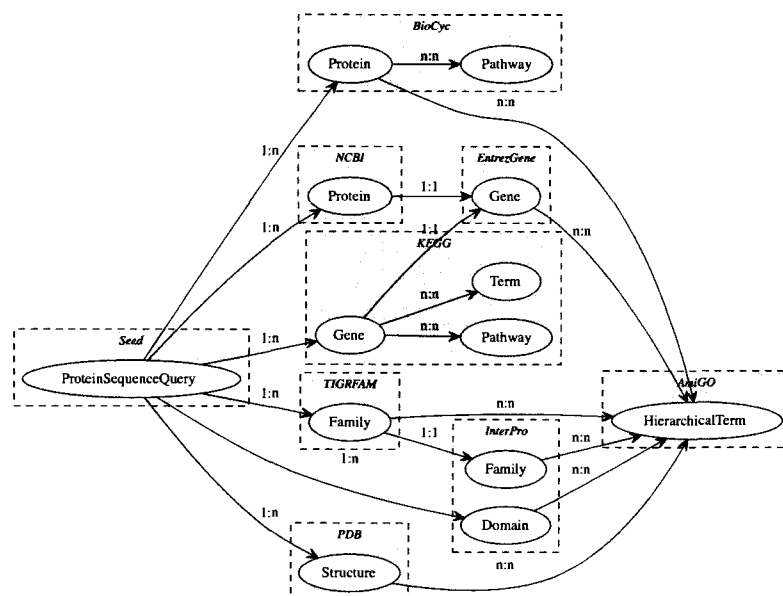


Figure 4.5: The above shows the data sources (in grey, dotted boxes), associated classes (ellipses), cross-links (arrows) and cardinalities (arrow labels) for the integrated data retrieval system. Queries begin on the left, with the *Seed* source, and are expanded rightward.

sources are valid for which queries, done through calls to the source catalogue and data directory. Additionally, the browsing engine manages how and when the *cache* is checked. Unlike other federated data integration systems, MIQAPS operates on a slightly hybrid model of data integration by caching data queried from live sources into a relational database. The purpose of this component is to save time in repeated queries; time-to-response from the cache is measured in milliseconds, whereas a very slow, computationally-heavy source may take a minute or more to return results. Data within the cache is stored simply, in a *key-value* pair, where the key is some label in a source interface and the value its contents. Thus, when data is returned from the cache, it appears to the schema, source catalogue and

data directory as if it was newly queried from a live source.

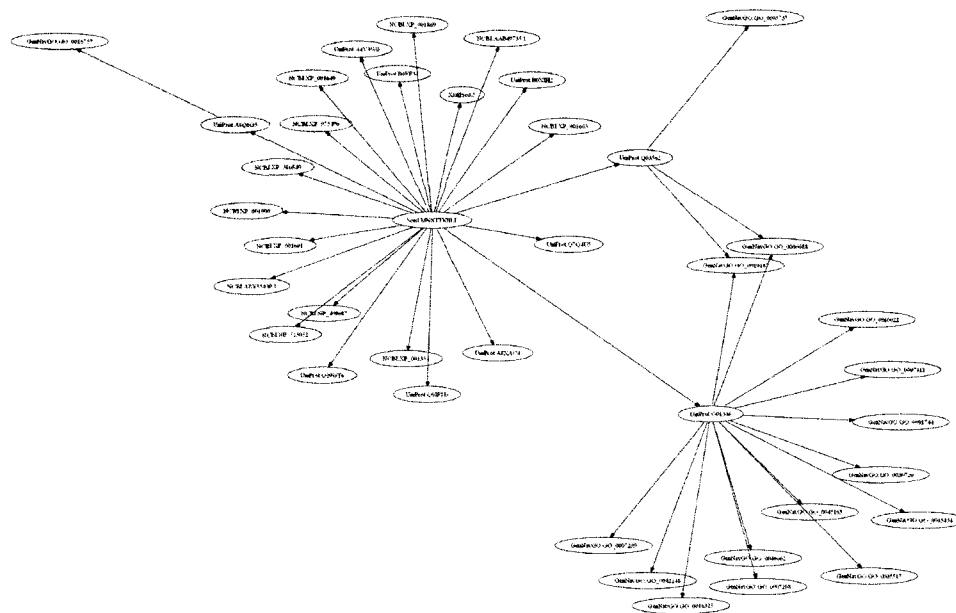


Figure 4.6: A small query graph generated using MIQAPS. Query graphs generated using MIQAPS are directed and acyclic.

The query graph is the conceptual representation of the interlinked and recursively queried data. In MIQAPS, the query graph follows the structure and form of the kind formally defined in Ch. 3.2.2. Each node in the query graph is the materialization of some entity in the schema, with information loaded from some source result, and each edge is the materialization of a link between two of these entities. In MIQAPS, edges generally represent one of two types: an explicit cross-reference from one source to another (*e.g.*, the referencing of a GO term from within a UNIPROT entry); or, a more uncertain link from one sequence alignment to another (*e.g.*, the connection between a protein sequence query and its BLAST results). Mechanisms exist within the system to allow the user to expand a single node, thus performing a selective join on a particular entity in the query graph, or expand all nodes indiscriminately.

Session management is also handled at the browser-level, and query graphs may be saved

and restored *via* serialization to a *Py DI Data Set*, or *PDDS*, file. PDDS files contain source results in the label-value format, and allow queries to be written to file, restored and then re-queried at a later time. The contents of the PDDS file are invariant to additive changes in schemata, such that if some schema (\mathcal{S}_A) under which a query graph was created is a literal *sub-schema* of some other subsuming *super-schema* (\mathcal{S}_B , so $\mathcal{S}_A \subset \mathcal{S}_B$), the query graph may be viewed and queried upon error-free under the super-schema.

4.1.3 Source interfaces

The layer between the data integration engine and the individual sources are the source interfaces, which execute the queries made system-side onto the remote databases. Concretely, these interfaces operate within the system as *generators*, whose input parameters are queries, and whose outputs are result *datasets*, the composition of which are records corresponding to individual results that have yet to be translated into schema entities. The virtual connection between the generators and the system are done *via* the `:gen` tag within the link declarations, as outlined in Ch. 4.1.1. In this way each link is aware of the necessary generator to call in order to produce the head-end of the join.

Separation of the source interfaces and the integration system itself produces a decoupled and flexible approach to managing changes to both how the data is modeled and remote sources are treated. Using this architecture, changes made to the sources would generally require commensurate changes only at the interface level, given that the data content itself is unchanged; mappings, linkages, entities and even queries can remain the same and still be valid, thus reducing effort over time. Practically speaking, as remote sources change in format, modifications would only be necessary to the individual generators that handle the sources.

The sources supported in MIQAPS are well-known biological repositories searchable *via* protein sequence, and provide coverage over several different biological domains [143, 144, 145, 146, 147, 148, 149, 150, 151, 152]:

- Genes, Proteins: ENTREZGENE⁷, ENTREZPROTEIN (*via* BLAST)⁸, BIOCYC⁹, KEGG¹⁰, UNIPROT¹¹
- Pathways: BIOCYC, KEGG
- Domains, Families: TIGRFAM¹², INTERPRO¹³, CDD¹⁴
- Structural data: PDB¹⁵
- Annotation terms: AMIGO¹⁶, GENNAV¹⁷

4.1.4 Support for structured queries

Exploratory queries form the information retrieval method of choice used throughout this dissertation, owing primarily to its inexpensive approach to gathering as much data regarding a query as possible. However, as a generalized data integration system it was essential to support the expression of more complex and specific queries in addition to the simple, albeit noisy, exploratory approach. Prior systems have benefited from the support of a formal query language layered over federated data, such as the BioNavigation platform and earlier versions of the BioMediator system [138, 153]. Importantly, these previous query language attempts have relied on the use of regular expressions as a means of validating paths along the resultant query graphs; queries were satisfied only when the paths satisfied

⁷<http://www.ncbi.nlm.nih.gov/sites/entrez?db=gene>

⁸<http://blast.ncbi.nlm.nih.gov/Blast.cgi>

⁹<http://biocyc.org>

¹⁰<http://www.genome.jp/kegg>

¹¹<http://www.uniprot.org/>

¹²<http://www.jcvi.org/cms/research/projects/tigrfams>

¹³<http://www.ebi.ac.uk/interpro>

¹⁴<http://www.ncbi.nlm.nih.gov/Structure/cdd/cdd.shtml>

¹⁵<http://www.rcsb.org/pdb>

¹⁶<http://amigo.geneontology.org>

¹⁷<http://mor.nlm.nih.gov/perl/gennav.pl>

the user-specified expression. Not all data is created equal, however, and for any individual biological task particular sources or entities may have higher ‘trust’ over others; these querying paradigms generally do not discern between the quality of data gathered from individual sources

Beginning with the successes in these other approaches to querying while at the same time attempting to address their shortcomings, a query language for MIQAPS was developed that supports the ability to pose ambiguously ranged queries that account for evidence and data coverage not explicitly supported in other query languages. Thus, in addition to basic exploratory queries, MIQAPS also supports queries posed in DaRQL (pronounced *dar-kle* and short for “*Domain-aware Regular Query Language*”). DaRQL is loosely based on SQL and SPARQL query languages used for relational databases and RDF graphs, respectively. Queries in DaRQL follow the following format outlined in Fig. 4.7.

```
TARGET < node_bindings >
FROM < start_node_bindings >
RESTRICT < path_and_node_constraints >
```

Figure 4.7: Basic structure of a DaRQL query.

In the figure, the **TARGET** clause provides a mechanism to allow a user to bind nodes which meet certain conditions to specific variables for later reference. The **FROM** clause anchors restrictions in the query with starting search nodes. Finally, the **RESTRICT** clause forms the selection of predicate constraints to enforce on satisfying nodes and paths, represented in regular expression form. These features are generally supported elsewhere in other systems that allow expressive queries over path-based data. However, an additional ability of DaRQL is explicit handling of nebulous queries of the type often seen in initial and formative biological research .

Figure 4.8 shows an example query in DaRQL¹⁸, with line 1 specifying two bindings such that all entities of type *Term* are bound to the variable ?t and all entities of type *Gene*

¹⁸For the complete grammar of DaRQL, refer to Fig. A.1.

```

1: TARGET ?t Term,?g Gene
2: FROM ?q Query
3: RESTRICT
4:     {?q.*.?g.?t},
5:     ?g:Species == "Homo sapiens" ,
6:     order(?t) > @MotifBased

```

Figure 4.8: Example query in DaRQL.

are bound to the variable `?g`. Line 2 specifies that the path search should begin at a *Query* entity, and lines 4-6 provide the constraints over valid paths. Specifically, line 4 denotes that a satisfying path should begin with a *Query* entity, have any number of entities following, and must end with a *Gene* entity with a outward edge to a *Term* entity. This is the format of path constraints in DaRQL, and follows a syntax similar to regular expressions. Line 5 indicates that any *Gene* within a satisfying path from line 4 has an attribute whose content is “Homo sapiens”, and thus enforces a requirement for species. Lastly, line 6 requires that all valid *Term* entities are supported by evidence whose rigor is greater than motif-based analysis.

The final line utilizes one of two special functions, `order`, in DaRQL that allow queries of ranged *evidence* and *coverage*. The other reserved function, `divcount`, allows the number range of diverse evidence types to be specified. Mappings between entities, sources and evidence types are handled in a separate evidence schema, or *domain hierarchy* which follows similar characteristics as the integration schema.

DaRQL queries are evaluated over pre-existing exploratory queries – that is, they are used as a means of curating and filtering pre-existing retrieved data. The query is handled by the query engine which manages the details of the path constraints and node bindings over the schema and the domain hierarchy. A query plan is then formed, which is optimized by restricting execution of the query to only parts of the graph which could possibly satisfy it. Once a plan is prepared, a series of *deterministic finite state machines* (DFSMs) are

created, each of which check the `RESTRICT` predicates over the graph by walking all paths. Thus, leaf nodes in the query graph which contain DFSMs that are valid form the end of a satisfying path that can then be serialized and presented to the user.

4.2 Motivations for design choices

While possessing many features found in other federated and path-based data integration systems, MIQAPS is unique in several ways that allow it to better handle high-throughput data integration for the purposes of generating data for statistical learning. As mentioned previously, unlike other biological federated data integration systems MIQAPS has been coupled with a relational database to allow efficient re-querying, an option that may be turned on or off at will. Additionally, the core representational components of the system, namely the schema, source definitions and edge mappings, are expressed using the same format. This is in contrast a system such as BioMediator, where the schema is represented in a mix of third-party software (Protégé¹⁹ for mediation, sources and edges) and an internal syntax (data directive files for mappings) [154]. An undesirable side-effect of this is while these data representation models are quite comprehensive, their areas of control tend to overlap, *i.e.*, management of mappings involves changing both the mediated schema and the mapping directives, and the directives require unwieldy server-related data in addition to the mapping data. In contrast, within MIQAPS, the models that manage the schema, data and mapping are cleanly separated from the implementation. Also unlike other federated data integration systems, MIQAPS may perform both exploratory queries and use structured queries for filtering and curation, the latter being carried out *via* a novel query language capable of data retrieval using parameters whose ambiguity towards evidence and coverage can be specified easily and expressively.

Further design choices were motivated by speed, both in execution and data processing, and long-term maintenance. Many systems rely on a standard interchange format both for external output of results as well as internal data transfer, such as XML or RDF. While this allows the data to be easily machine-readable and structured generically, such

¹⁹Protégé ontology editor, <http://protege.stanford.edu>

formats tend to be verbose and involve non-trivial overhead processing. Internally, all data within MIQAPS is stored in tuple form (*label-value* pre-translation, and entity tuples post-translation, within the data directory) which can be quickly processed, at the cost a generalized data exchange format; externally, results in MIQAPS may be serialized in XML and thus inter-system information exchange is preserved. The inclusion of a cache within the system further allows expedited retrieval of data, provided those results have been queried and returned successfully in the past.

Finally, the choice of Python as the implementation language was driven primarily by the desire for a succinct and thread-friendly scripting language. While a language such as Perl [155] is more recognized and widely adopted within the biomedical research community, it was important to select an environment with full-featured and mature object-oriented capabilities. At the same time a language with little development overhead was desired, given the nature of how biological resources evolve thus necessitating semi-regular updates to the interface layer. Java [156], though it is a language that has been used extensively in the past for various components of data integration systems [35, 138, 157], often requires extensive overhead and source maintenance that most small biological laboratories would be hard-pressed to upkeep.

System	Notable characteristics					Prime service
	Architecture	Data model	Model repr.	Exchange format	Query method	
Atlas [33]	Warehouse	Relational	SQL	Internal	Programmatic	Infrastructure
BioGuideSRS [34]	Federation*	Ontology	RDF	RDF/OWL	Graphical	Guided queries
BioMediator [35]	Federation*	Mediated schema	Protégé frames	XML	Graphical/programmatic	User-centric queries
BioNavigation [36]	Federation*	Ontology	Regular expr.	Internal	Reg. expr.	Guided queries
BioZon [37]	Warehouse*	Hierarchy	SQL	Internal	Preconstructed	User-centric queries
K2/Kleisli [38]	Federation	Nested relational	ODL/CPL repr.	CPL repr.	OQL	Infrastructure
MIQAPS [158]	Federation*	Mediated schema	S-expression repr.	Internal	DaRQL	Infrastructure/guided queries
OPM [39]	Federation	Object-oriented	Internal schema	Internal?	OPMQL	Infrastructure
SRS [40]	Federation	Linked flat-files	ODD language	Internal	Preconstructed	Indexing
TAMBIS [41]	Federation	Description logic	GRAIL repr.	CPL repr.	Preconstructed	User-centric queries

Table 4.1: A comparison of biological data integration systems along selected features. An asterisk (*) indicates that the integration architecture supports path-based queries. *Guided queries* refer to systems with sufficiently expressive query languages that the query path may be parameterized; *user-centric queries* refer to systems geared toward exploratory queries; and systems denoted with *infrastructure* are those that are explicitly designed to be layered with other tools, and are not necessarily stand-alone *per se* [33, 34, 35, 36, 37, 38, 39, 40, 41].

4.3 Discussion

This chapter has provided the design and implementation details of a federated data integration system with a schema and sources tuned specifically for the querying and retrieval of information related to protein sequences for the purpose of annotation. Though such data integration systems are not new, several features have been discussed that make MIQAPS uniquely suitable for dealing with high-throughput sequence queries posed both in an exploratory and structured fashion.

Chapter 5

CHARACTERIZING INTEGRATED ACCESS TO BIOLOGICAL INFORMATION

The previous chapter outlined a method and system of integrating fractured and heterogeneous data *via* path-based federation. The effectiveness for path-based federated data integration relies on the interconnected nature of the databases of interest in order to retrieve a large amount of relevant data that would otherwise be difficult to gather one source at a time, one query at a time. However, data within these databases can be of questionable quality and highly speculative. By their nature, exploratory queries on this data will provide results of high recall and low precision – there may be significant *noise* in retrieved results in the form of spurious data.

This chapter¹ comes after the introduction of the MIQAPS system from the previous chapter, and describes tests on the framework for the retrieval of high-quality information from biologic databases. These experiments rely on a fault tolerance model of reaching ‘trustworthy’ information to test the use of naïve exploratory queries for function identification, the findings of which help in understanding the cross-referential characteristics of biological databases for annotation. The revelation is that data sources share a considerable amount of redundancy, for both high-quality data and probable noise; using basic and exploratory data integration methods alone, it would be difficult for a scientist to discern between the two consistently.

¹The related work, methods, results and following illustrative elements of this chapter appear in a prior publication by Cadag, Tarczy-Hornoch and Myler entitled “On the reachability of trustworthy information from integrated exploratory biological queries”, in *Lecture Notes in Computer Science* (Data Integration in the Life Sciences 2009) 5647 pp. 55-70 with kind permission of Springer Science+Business Media: Figs. 5.2, 5.3, 5.4, 5.5, 5.5 and 5.7; Tbls. 5.2 and 5.3.

5.1 *Reachability of trustworthy integrated data*

As was seen in Ch. 2, whether annotation is done manually or using automated methods, functional annotation of a protein is often done using multiple, corroborating data sources, and an annotating scientist will generally rely on a variety of different and diverse biological databases before reaching a final conclusion [54]. The implications of encountering noise when integrating data for sequence annotations are obvious, since data of low quality will invariably lead to poor functional analyses. Subsequently, a simulation of interest would be to determine the likelihood that an annotator would reach a high-quality and correct annotation for a protein query as they follow references from one data source to another. Under this hypothetical scenario, the “naïve annotator” may choose to stop searching at any given time.

While simplified, this basic approach is not unlike how functional annotation is done manually, an often *ad hoc* and idiosyncratic process [19]. Determining how well an annotator would reach a high-quality annotation would provide insight into the use of data integration for biological research. Clearly, if the signal to noise ratio provided by data integration was low, then it may behoove a user to focus efforts more on navigating the most useful data source. On the other hand, if the value added by data integration for a task such as annotation was high, then the benefits of coverage and redundancy associated with data integration systems may be worth the extra effort of parsing and analyzing data from disparate sources.

Using data collected *via* a federated, path-based framework, the remainder of this chapter discusses the methods, results and analyses of just such an annotation simulation, based on the use of a common data model built for multi-source triangulation and employing the MIQAPS system as a proxy annotator. Roughly approximating methods and sources used by scientists for annotation [54, 19], it was found that unverified information of lower provenance was almost equally as encountered as experimental results, largely due to the amount of redundancy present in the databases; signal and noise is often indistinguishable by reachability alone.

5.2 Related work

Others have conducted various studies in the past to analyze networks of cross-referential data. For example, Lacroix *et al.* leveraged graph statistics such as path length and cardinality to aid in efficient query planning over graphs of biomedical data [134]. Additionally, Louie *et al.* used a random sampling model similar to that described later in this chapter to determine individual belief probabilities in the relevancy of integrated data records [104]. The question addressed here is not for the purposes of relevancy ranking or query planning *per se*; rather, the problem of interest is in measuring the accessibility of correct biomedical data using data integrative methods, and whether or not data integration by itself is a necessary and sufficient approach for triangulation of high-quality annotations.

Research has also been done on naturally-occurring networks. Methods of reachability and connectivity in these instances have been valuable in determining the robustness of interaction pathways in model organisms, for example. Albert *et al.* demonstrates that a particular type of graph, *scale-free networks*, are particularly resistant to random failure when compared to binomial networks, and Amaral *et al.* showed that neural connectivity showed similar signs of tolerance [159, 160]. In a somewhat related previous study, Searls showed that database schemata and software dependencies display similar robustness, at least in part [161]. Methods such as these for evaluating network tolerance are adapted in this dissertation for the purposes of testing biological data reachability in the face of increasing failure rates.²

5.3 Methods

5.3.1 Fault tolerance model over arbitrary query graphs

The “naïve annotator” scenario described in the previous section can be simulated in a straightforward manner by conducting *random walks* over arbitrary graphs. When these graphs are query graphs of integrated data, the effect is akin to functional annotation,

²The text of Ch. 5.2 has been adapted from a previously-published paper by Cadag, Tarczy-Hornoch and Myler entitled “On the reachability of trustworthy information from integrated exploratory biological queries”, in *Lecture Notes in Computer Science (Data Integration in the Life Sciences 2009)* 5647 pp. 55-70 with kind permission of Springer Science+Business Media.

given the query of interest is an unannotated protein. This basic approach can be modeled stochastically and the reachability of particular ‘annotations’ can be assigned probabilistic likelihoods. Formally, for a query graph G seeded with a protein sequence and generated using the data integration methods described in Ch. 4, the task at hand is to measure how well connectivity holds from a seed node $s \in V$ to some other node $t \in V$, which for annotation purposes may be a GO term. The reachability problem, otherwise known as *st-connectivity*, is \mathcal{NL} -complete and, *via* random sampling, likelihood estimates $s \rightsquigarrow t$ ³ under arbitrary conditions (*i.e.*, weighted nodes, edges) can be determined.

Let $R \in V$ represent the nodes of interest for which to test reachability from s (R may be the set of experimentally validated data, for instance), and l a specified failure rate – the probability that any random node $n \in V \setminus (R \cup \{s\})$ ⁴ is unavailable, whereupon n and edges entering and leaving n are removed from G , recursively. This model emulates instances where a data record did not return as expected, due to source downtime, retrieval processing problems or errors of omission on the part of the scientist. Denote the event of removing nodes as inducing node *failure* F . Then define the probability that any node $t \in R$ retains a path from s after F as:

$$p(G, R, l) \equiv P(s \rightsquigarrow t | F_l). \quad (5.1)$$

The algorithm in Fig. 5.1 executes the random and recursive node removal process, thus generating an estimate of the probability expressed in (5.1).

Extending (5.1) to all possible failure rates and taking the area under the resulting curve gives the *fault tolerance* of a graph G given arbitrary nodes R with respect to l . Formally, the fault tolerance, τ , of a query graph is expressed by:

$$\tau(G, R) = \int_{-\infty}^{\infty} p(G, R, l) dl. \quad (5.2)$$

Note that as $p(G, R, l)$ defines a probability function, $\tau(G, R)$ is bounded from below by 0 and above by 1. In practical terms, one can imagine τ as the probability that an annotator,

³Where $s \rightsquigarrow t$ denotes that a transitive path exists from s to t .

⁴Where $A \setminus B \equiv \{x \in A | x \notin B\}$.

following links from biological data source to data source and who may choose to abandon a path or select one path over another, will reach any term within a pre-specified set of GO terms R .

Algorithm 1: Estimating query graph tolerance at a specific failure rate

Input: $G = (V, E)$ with initial query $s \in V$, targets $R \subset V$, simulations $i \in \mathbb{Z}^+$, failure rate $l = (0, 1)$

Result: tolerance value $\tau = (0, 1)$ for G, R w.r.t. failure rate l

```

1  $\mathbf{m} \leftarrow []$ 
2 for iteration  $j, i$  times do
3    $H = (V', E') \leftarrow G, S \leftarrow R$ 
   // Sample random nodes for removal (excluding the provided targets and seed)
4   for  $|V'| * l$  times do  $V' \leftarrow V' \setminus \text{sample}(V' \setminus (R \cup \{s\}))$ 
5   for  $t \in S$  do
6     if  $\neg(s \rightsquigarrow t)$  in  $H$  then  $S \leftarrow S \setminus \{t\}$ 
7   end
8    $\mathbf{m}_j \leftarrow \left(\frac{|S|}{|R|}\right)$ 
9 end
10 return  $\frac{1}{i} \sum_k \mathbf{m}_k$ 

```

Figure 5.1: Sampling procedure used to approximate $p(G, R, l)$ with respect to target nodes R , in query graph G , at a failure rate l ; applying this with $l = (0.0, \dots, 1.0)$ yields $\tau \approx \frac{1}{|l|} \cdot \sum_{k \in l} p(G, R, k)$.

Denote the tolerance curve of query graphs hereafter as τ_{exp} , with the above R the set of experimentally determined GO terms. It is also desirable to generate the *random tolerance curve* $\tau_{rdm}(G, N)$, where $N \subset V \setminus \{s\}$ and $|N| = |R|$, with the latter property used when in comparison with τ_{exp} . Calculation of τ_{rdm} would proceed in a similar way as τ_{exp} with the exception that nodes in N may be chosen randomly and at each iteration, and allow the possibility that nodes in R and any given N may overlap. The random tolerance curve provides a baseline level of average reachability for G with which to compare to the experimental tolerance curve, and N may be set to a specific set of nodes in a particular domain, such as when comparing well-curated GO terms, R , with poorly curated ones, placed in N .

In the context of integrated biological data sources, this allows the measurement of how

well-curated data is connected to an original query *via* any number of paths and links of questionable relevancy. For example, data that is known *a priori* to be of higher-quality that retains reachability from s in the face of random failure, while at the same time lower-quality data loses reachability, suggests a query in which one can assign a reasonable level of confidence that sound results can be easily separated from unsupported results. On the other hand, graphs where lower-quality data retains reachability from the query better than high-quality data imply that separating the “wheat from the chaff” may involve a great deal of effort. As the above method is based on simple random sampling, it can easily be adapted to weighted edges, or examinations focused on subparts of a query graph, such as inducing failure only upon nodes that originate from a particular source; this enables explorations of data reachability in the face of specific source unavailability or omission, situations not uncommon for federated data integration systems.

5.3.2 *Estimating tolerance of biological queries for high-quality data*

The model formalized in Ch. 5.3.1 was applied to query graphs generated by MIQAPS in the form of an external module whose inputs were saved query graphs and whose outputs consisted of τ values at various failure rates. The implementation of MIQAPS used is outlined in Fig. 5.2. Exploratory queries posed to MIQAPS for tolerance calculation were automated, and user involvement was limited to the submission of a single FASTA file containing all sequences.

For the purposes of this experiment, a small number of UNIPROT accessions from the GENE ONTOLOGY ANNOTATION DATABASE⁵ (GOA) UNIPROT which contained GO annotations made on the basis of experimental evidence, as outlined *via* GO evidence codes⁶ in GOA (*e.g.*, ‘EXP’, ‘IGI’) were randomly selected. GO codes annotated in this fashion were considered to be experimentally verified, while all others (*e.g.*, ‘ISS’, ‘TAS’) were nominally treated as not experimentally validated (non-verified). Thus, within an actual query and for evaluation purposes a GO term may be labeled as experimental, as determined by GOA,

⁵<http://www.ebi.ac.uk/GOA>

⁶GO evidence codes at <http://www.geneontology.org/GO.evidence.shtml>

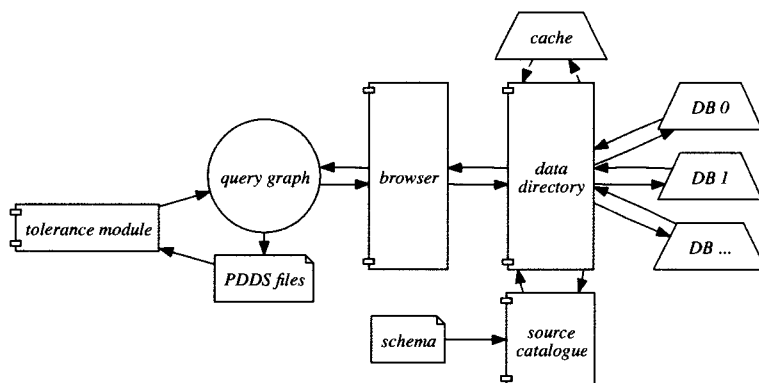


Figure 5.2: MIQAPS architecture, adapted for fault tolerance experiments. The *tolerance module* enacts the simulation over saved query graphs generated using exploratory sequence queries.

even if the associated evidence code in the query graph itself is not experimental. 104 proteins were identified whose annotations included experimental codes, and which returned GO terms when queried using MIQAPS. Per Figure 4.5, seven different data sources were used to reach GO terms from the initial sequence query.

Caveats to the approach described above include the possible confounder that annotations linked from one of the sources MIQAPS covers contains GO terms transitively assigned from GOA UNIPROT. This was somewhat mitigated by exclusion of UNIPROT from the list of sources used; unfortunately, this problem is difficult to avoid when dealing with protein annotations across databases that are often referenced to curate other databases. Moreover, though GOA-determined experimental GO terms were treated as a evaluation standard and the sole measure of high-quality data, many GO terms that were not experimentally verified may have been equally valid.

Because estimation of (5.1) is based on a simple random sampling approach that requires a large number of simulations, initial testing was done for the optimum choice of simulations that was not overly time-expensive but still converged on a value for τ . Using 11 randomly-

<i>GO Code</i>	<i>Description</i>
EXP	Inferred from experiment
IDA	Inferred from direct assay
IPI	Inferred from physical interaction
IMP	Inferred from mutant phenotype
IGI	Inferred from genetic interaction
IEP	Inferred from expression pattern

Table 5.1: Experimental Gene Ontology evidence codes, from <http://www.geneontology.org>.

selected proteins from the set of 104, it was found that on average convergence began at approximately 10 000 simulations (see Figure 5.3); this was thus the value used for i in running the algorithm in Fig. 5.1.

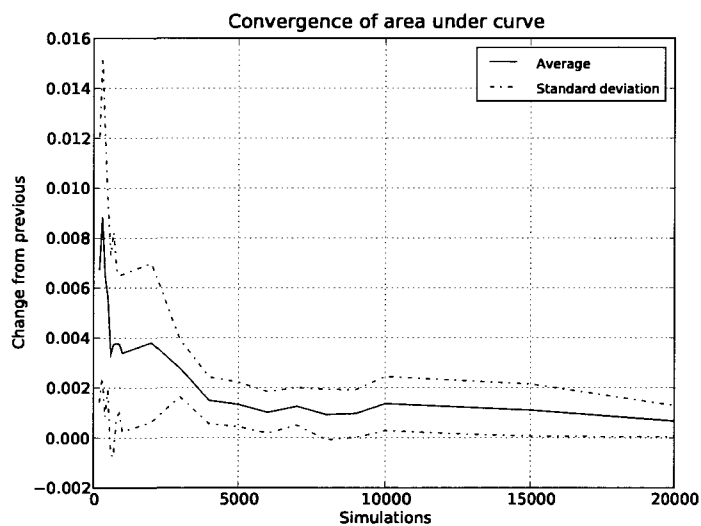


Figure 5.3: Based on 11 randomly selected queries, convergence of the fault tolerance measure τ started at approximately 10 000 simulations.

In addition to comparing the behavior of the tolerance curves and fault tolerance values

τ_{exp} and τ_{rdm} in the form of $\tau_{exp-rdm} \equiv \tau_{exp} - \tau_{rdm}$, general graph statistics were also examined, such as the number of nodes and edges, the radius, the number of nodes originating from each source and the number of recognized *a priori* experimental GO terms, among others. The probability that a GO term t is experimentally verified, given that there exists a node from a non-AMiGO source D which has a path to t , *i.e.*, $P(t \in R|D \rightsquigarrow t)$ was also obtained, and was readily calculable from the query graphs themselves.⁷

5.4 Results

Using the methods described in Ch. 5.3.1 and Ch. 5.3.2, τ_{exp} and τ_{rdm} curves were generated for each of the 104 sample proteins. Overall, τ was greater for the experimental GO terms than random GO terms in 60.5% of the cases, which implies that curated and verified annotation data will tend to be referenced more often in data sources than non-verified annotations. On closer detail, however, there was notable variety in how well an individual query was connected to a GO term through the data sources; values for τ_{exp} ranged from 0.348 to as high as 0.882, and τ_{rdm} 0.348 to 0.790. Fig. 5.4 shows four different tolerance curves from the test set, representative of the curves generated overall; τ_{exp} ranges from extremely tolerant against node failure to slightly-worse than the proportional failure rate.

The variation is made clearer when the average curves of τ_{exp} and τ_{rdm} are compared across all 104 proteins (see Fig. 5.6). The average difference between the two curves is only marginal, although τ_{exp} is significantly more varied towards the upper end. A possible explanation for this is that there are a minority of proteins where the curation toward experimentally verified results has been more thoroughly propagated throughout many of the data sources, and so there are numerous paths to the term from the query. In this group, even with as high as a 50% node failure rate across the query graph, a number of experimental GO terms remain reasonably reachable. For other proteins, however, it is difficult to discern the experimentally verified GO terms from others based on data source cross-linking alone.

⁷The text of Ch. 5.3 has been adapted from a previously-published paper by Cadag, Tarczy-Hornoch and Myler entitled "On the reachability of trustworthy information from integrated exploratory biological queries", in *Lecture Notes in Computer Science (Data Integration in the Life Sciences 2009)* 5647 pp. 55-70 with kind permission of Springer Science+Business Media.

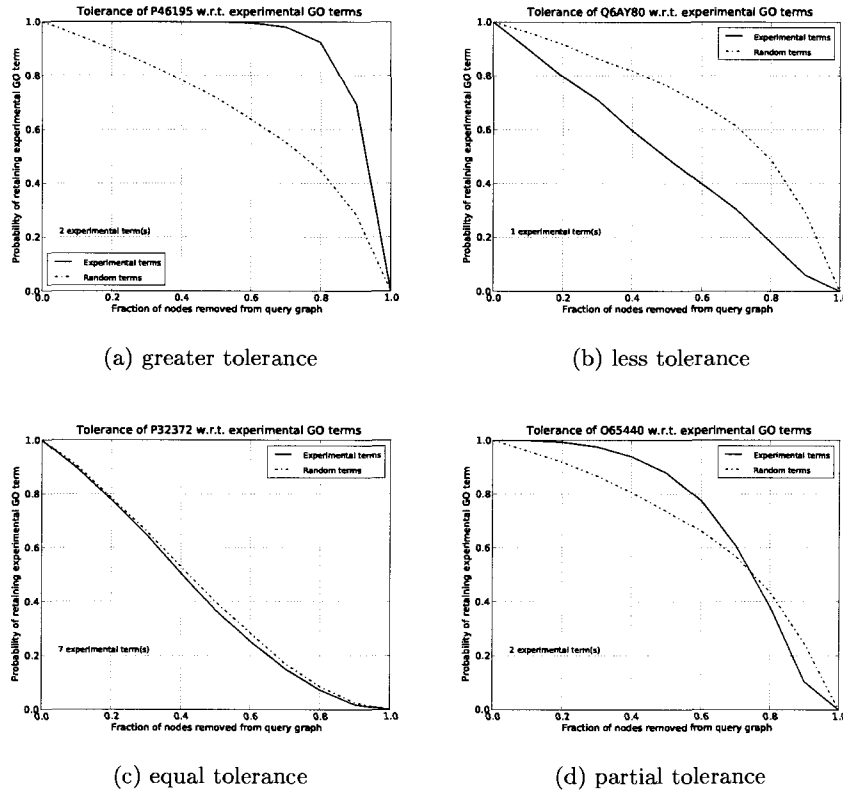


Figure 5.4: Selected archetypal tolerance curves (τ_{exp}) for experimentally validated Gene Ontology terms compared to non-experimental (τ_{rdm}).

Fig. 5.5 shows the distribution count of $\tau_{exp-rdm}$ for the GOA protein set. Again, it is clear that many experimental terms are generally indistinguishable from random terms in regards to reachability, as the number of overall proteins where $\tau_{exp-rdm} > 0$ is slight. However, using a two-tailed paired non-parametric test, it was found that τ_{exp} and τ_{rdm} in fact are derived from significantly different distributions, and this suggests that despite the visual parities found in the tolerance curves generally, experiment terms are more reachable, from a statistical standpoint ($p\text{-value} = 0.023$, statistically significant at $\alpha = 0.05$).

To explore this further, the graph statistics of the top 10 and bottom 10 query graphs, as sorted by $\tau_{exp-rdm}$ in Tbl. 5.2, were examined. Interestingly, the query graphs where

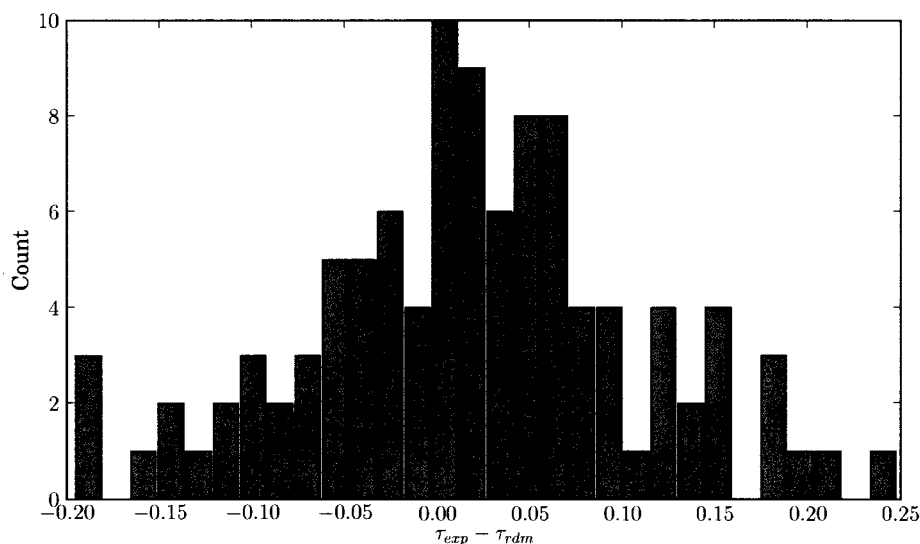


Figure 5.5: Histogram of $\tau_{exp} - \tau_{rdm}$ for the test proteins. Values below 0 indicate that random GO terms are more fault tolerant to node loss, and thus more reachable, than experimental GO terms, and *vice versa*.

$\tau_{exp-rdm}$ is greatest were smaller than those where $\tau_{exp-rdm}$ were least, in terms of absolute number of edges and in ratio to the number of nodes (2.80 edges/node versus 2.34); the queries where the experimental results were most accessible were less connected than those where the experimental terms were indistinguishable from random terms, path-wise.

In terms of biological relevance, the queries with a large $\tau_{exp-rdm}$ tended to be entries in UNIPROT that were reviewed (70%), half of which originating from the mouse proteome. Conversely, queries where $\tau_{exp-rdm}$ were most negative were more likely not to be reviewed (40%), and come from a more phylogenetically diverse list of organisms. As the mouse model is commonly used in biomedical research, the greater likelihood of arriving at an experimentally-based conclusion in the former is not surprising. Indeed, high $\tau_{exp-rdm}$ queries contained more experimental GO terms in their query results than low $\tau_{exp-rdm}$ queries, although there was no strong correlation between $\tau_{exp-rdm}$ and the number of experimental GO terms among the 104 queries overall.

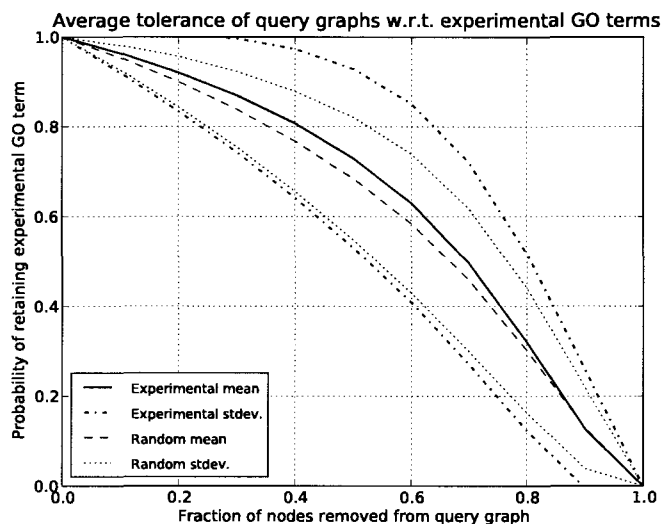


Figure 5.6: Average tolerances and standard deviations between experimental GO terms and random GO terms.

Results thus far have been in the case of *random* failure – that is, any node from any source having a fractional l chance of being lost, much like how an annotator may miss or choose to ignore pursuing links while characterizing a protein. In order to measure data coverage, the effect of omissions of entire sources was analyzed. To this end, each of the sources that link to GO terms individually were systematically removed along the same linear failure rate $l = (0.0, \dots, 1.0)$ and the effects incremental loss of individual sources had on tolerance were reviewed.

Tbl. 5.3 shows the results of individual source unavailability; τ is significantly higher in these instances as only a single source at a time was removed, and the rest of the query graph remains untouched. It immediately stands out that in the case of all but one of the sources (ENTREZGENE), the query graphs generally retain connectivity to GO terms, both random and experimental. Loss of ENTREZGENE does reduce τ significantly; however, ENTREZGENE nodes constitute on average 43% of the nodes in the query graphs that link to GO terms, yet have an impact of only 27-29% when removed, which lends credence to

<i>Accession</i>	$\tau_{exp-rdm}$	<i>Species</i>	<i>Reviewed?</i>
Q9JIL4	0.249	<i>M. musculus</i>	×
Q49IK6	0.242	<i>D. rerio</i>	
P46195	0.225	<i>B. taurus</i>	×
O88379	0.199	<i>M. musculus</i>	×
Q61176	0.184	<i>M. musculus</i>	×
O55081	0.175	<i>R. norvegicus</i>	×
P55209	0.173	<i>H. sapiens</i>	×
Q14AC4	0.165	<i>M. musculus</i>	
Q10176	0.154	<i>S. pombe</i>	×
A2A602	0.148	<i>M. musculus</i>	
Q2KT22	-0.194	<i>D. rerio</i>	
Q19328	-0.192	<i>C. elegans</i>	
Q8AY90	-0.183	<i>D. rerio</i>	
Q6AY80	-0.178	<i>R. norvegicus</i>	×
A4FVK4	-0.175	<i>D. rerio</i>	
P46974	-0.142	<i>S. cerevisiae</i>	×
P91621	-0.127	<i>D. melanogaster</i>	×
Q24133	-0.111	<i>D. melanogaster</i>	×
B3DFT2	-0.104	<i>D. rerio</i>	
Q7JLC3	-0.093	<i>C. elegans</i>	

Table 5.2: Top 10 and bottom 10 in $\tau_{exp-rdm}$; proteins where the experimental GO terms are likely to stand out are from mammalian proteomes, or have been reviewed.

the conclusion that there is considerable redundancy and overlap amongst the sources.

Examining the probabilities that a source will lead to an experimental GO term, given that it leads to any GO term, provides additional information with regard to the overlapping coverage between the databases. For example, $P(t \in R|D \rightsquigarrow t)$ is higher in KEGG than in ENTREZGENE, yet incrementally removing KEGG decreases the likelihood of encountering random, non-experimental terms, whereas the reverse is true for ENTREZGENE. A hypoth-

<i>Data source (D)</i>	<i>Exp. GO terms (τ_{exp})</i>	<i>Rand. GO terms (τ_{rdm})</i>	<i>$P(t \in R D \rightsquigarrow t)$</i>
BIOCYC	1.000	0.966	0.052
ENTREZGENE	0.710	0.729	0.164
INTERPRO	0.999	0.994	0.101
KEGG	1.000	0.994	0.167
PDB	0.998	0.969	0.053
TIGRFAM	0.993	0.994	0.070

Table 5.3: Average query graph tolerances for experimental Gene Ontology terms with source-targeted node failure, and likelihoods that a term is experimentally derived, given that a source has a path to it.

esis to explain this occurrence may be that as a source, KEGG is likelier to reference GO terms to which other sources already link, and thus is useful for redundant coverage, but is less effective at uncovering novel, experimental data. ENTREZGENE, on the other hand, appears to be a rich source for experimental GO terms, and although it had relatively few links to GO terms by comparison to make a significant impact, TIGRFAM possibly shares similar attributes.

Finally, the overall graph structure of query graphs was compared with well-known graph structures of similar composition – namely, binomial (Erdős-Rényi) graphs, and power law graphs – with the intent to measure how graphs built from cross-references of disparate sources compare in resiliency to other graphical models. Representations of the canonical graphs were generated such that the node and edge counts, likelihood of edges between nodes and size $|N|$ were similar to query graph averages. Seed nodes in these graphs were simulated by randomly selecting a node in the center of the graph, and random nodes on the periphery were chosen as ersatz “GO terms.” (see Fig. 5.7).

Not surprisingly, binomial graphs lost connectivity to the peripheral nodes rapidly, as edges were formed independently and the formation of strongly-connected clusters or hubs was unlikely (incidentally, a number of individual query graphs appeared to follow this model

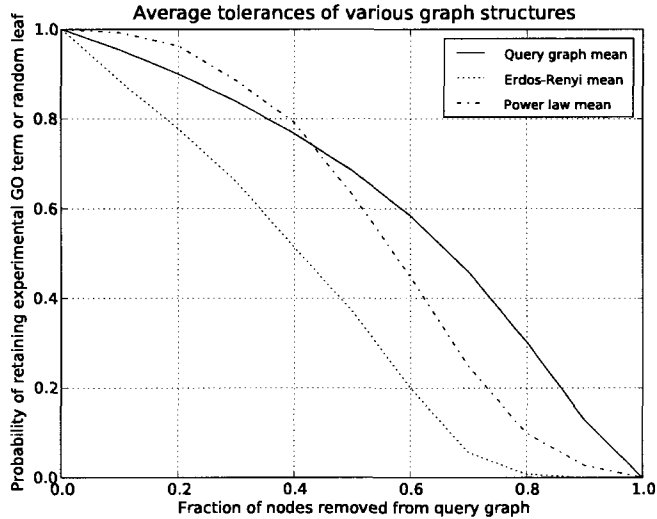


Figure 5.7: Comparison of the random tolerance curve from query graphs, to the tolerance of similarly-composed power law and Erdős-Rényi graphs. Above, $\tau_{rdm} = 0.601$, $\tau_{pow} = 0.553$ and $\tau_{erg} = 0.406$.

as well). At the opposite end of the spectrum, the power law graph maintains resiliency over the average query graph for $l \leq 0.45$ before it is overtaken. This is an interesting finding, as it suggests that some query graphs appear to have properties of resilience equal to or greater than that of scale-free networks. Generalizing the resiliency of query graphs beyond the narrow seed-to-term focus as it was studied here may be specious, however, since the sources and references used here were specifically geared towards forming paths to a targeted set of nodes.⁸

5.5 Conclusion

The findings for the use of tolerance curves as a means of estimating reachability of high quality biological annotations seem to both validate and complicate the view of how biologi-

⁸The text of Ch. 5.4 has been adapted from a previously-published paper by Cadag, Tarczy-Hornoch and Myler entitled “On the reachability of trustworthy information from integrated exploratory biological queries”, in *Lecture Notes in Computer Science* (Data Integration in the Life Sciences 2009) 5647 pp. 55-70 with kind permission of Springer Science+Business Media.

cal information is cross-referenced in public databases. The results reported in the previous section suggest that there is indeed a substantial amount of noise in these databases, and that for any given protein it can be difficult to determine whether or not the most frequently-encountered annotation is correct. That does not, however, omit the finding that there are indeed cases where the most frequently-encountered annotations are also the most well-curated and thus of greater value than average. The fact that the distributions of τ_{exp} and τ_{rdm} are different implies that there may be ways of augmenting naïve data integrative methods in such a way as to optimize the selection of annotations for at least a subset of proteins.

Furthermore, while the model of failure measurement described in Ch. 5.3.1 is theoretically robust enough to handle more complex and rich graphical representations, such as weighted edges (which are, incidentally, akin to what one might find naturally, *e.g.*, expect values between aligned sequences, where the sequences are the nodes), the approach used here does not take advantage of this. As such, the results represent somewhat of a lower bound of the performance of data integration – retrieval and traversal of data that is indifferent to the quality of the paths and nodes of interest. For experiments and discussions on the topic of weighted graphs, refer to work by Louie *et al.* who use related methods to measure computational graph-based methods against manually curated functional annotation (as opposed to experimental, as was done here) [104]. These simulations were also conducted using only exploratory queries. Further work would be required to test if the conclusions on reachability hold when targeted and specific queries are used; this is still an area of open research.

5.6 Discussion

This chapter, by using tests of reachability oriented around fault tolerance, has demonstrated that while some high-quality data dominates in a select number of proteins, it is overall difficult to discern trustworthy data (interpreted as experimental evidence) from less trustworthy data for annotation across biological data sources. Though annotation noise is a well-published facet of molecular biology databases, what is surprising is the difficulty with which to distinguish even very well-curated, experimental data from model organisms,

as seen in the results. From these findings, one possible conclusion is that data integration in itself may be insufficient to properly parse apart signal from noise amongst biological data. The implications of this are far-reaching, as integrative methods both *ad hoc* and formal form the backbone of many automated functional annotation methods in biology. In the face of these challenges, complementary methods are needed to address the shortcomings produced by massive retrieval of biological data that is of varying relevance.

Chapter 6

**LEARNING PATHOGENIC PROTEINS FROM INTEGRATED
QUERY NETWORKS**

The research of this dissertation rests on two main premises, the first being that the resulting path-generated graphs of integrated queries across multiple, fractured data sources add value to the retrieved data *via* triangulation and coverage, and that the resulting graphical structure of these integrated queries can be used in a predictive capacity to determine characteristics of the query itself. The information retrieved in this way and cross-referenced using formal integrative methods is vast and has strong potential for discovering biomedically-relevant relationships. At the same time, as shown in Ch. 5, mutual references between databases is insufficient for gleaning high-quality data. This chapter¹ discusses the methods used in conjunction with the MIQAPS system (as outlined in Ch. 3) to identifying general virulence proteins² using a variety of statistical learning approaches.

6.1 Methods*6.1.1 Learning query graphs*

Presumably, within a path-based model, the closer a node is to the initial query, the more relevance it has; those further from the query itself are multiple sources detached from the direct query and are theoretically of waning relevance. The query graphs used in this research were fully grown (*i.e.*, all expansions were exhausted) according to the mediated schema, and many graphs include records that may be quite distant from the initial query. Prior work by others in the field of biologic data representation explored various methods

¹The methods and results of this chapter appear in an as-yet unpublished manuscript by Cadag, Tarczy-Hornoch and Myler entitled “Path-based integration of heterogeneous biological data for learning bacterial virulence proteins”.

²The virulence data used in this chapter was graciously prepared and formatted for use by Jason Smith at the Lawrence Livermore National Laboratory’s Pathogen Bioinformatics Group, to whom the author is most grateful.

of exploiting the graph structure for inferring the relevance of individual nodes and paths. Bharat and Henzinger, for example, describe several algorithms, such as ones that use the in- and out-degree of nodes, for the analogous problem of determining topical relevance of hyperlinked documents [162]; Tsuda *et al.* use a diffusion-based approach to assign weights within protein networks [163], a method readily adaptable to query graphs; Weston *et al.* apply a rank propagation algorithm on sequence similarity graphs generated from PSI-BLAST hit values [164]; and Detwiler *et al.* test a variety of methods, such as Monte Carlo simulations and relevance propagation, to rank nodes in similar graphs [20]. Here, the interest is less in the relevance of any individual node; rather, the interest is in the value of using the graph globally and as a representation of the query in classification activities.

After seeding an initial query, retrieving records and fully growing the query graph, the next step was to transform the content of the graph into a representation more amenable to classification. The technique used to accomplish this involved weighting nodes in the query graphs; query graphs become readily transformable into a feature vector representation, at which point various statistical learning methods can be employed. Doing so bypasses the difficulties in comparing query graphs directly, and depending on the weighting scheme used can still leverage the benefits of the graph structure. Because the query graph is generated by a series of cross-linkages across databases, nodes that are cross-linked most often or have strong sequence-similarity to the query can be weighed highest. An ideal scheme would heavily weigh nodes that characterize the query sequence more precisely, and lightly weigh nodes that do not, thereby minimizing noise.

This work relied on methods similar to that of Detwiler *et al.* and Chua *et al.* as this approach was found to be among the computationally least-expensive while still performing well [20, 18]. The subjective weights used by Detwiler *et al.* assigned to each data source, link and record determined *a priori* were omitted in this case. The belief was that as the targets of interest are the query graphs the correlation of the weights to the actual relevance requires only a modicum of precision, since query graphs are treated as classifiable representations and not individual results within the graphs *per se*.

Begin by letting $w_t(n)$ be the weight of node n in the query graph at iteration t , and

that:

$$0 \leq w_t(n) \leq 1, \text{ for all } t, n. \quad (6.1)$$

To take advantage of the interconnected nature of the query graph, it is desirable to allow $w_t(n)$ to be influenced by its neighbors, recursively. This represents the grounding that a user's posed query is the most confident node within the graph, and that further confidences emanating from resultant queries are derivative of this, and *propagate* outwards. A natural way of representing degradation of confidences between nodes in an exploratory, query sequence-based graph would be expect values, such that $expect(p, n)$ is a function which returns the expect value from some query p to the result n . Thus, the influence of a node's inward-joining neighbors may be represented as a factor of both those neighbors weights ($w_{t-1}(p)$) and their relation to the target node ($expect(p, n)$), *i.e.*, :

$$w_t(n) \propto \prod_{(p,n) \in E} w_{t-1}(p) \psi(p, n), \quad (6.2)$$

or, under the mathematically convenient assumption that the right-hand of (6.2) defines a probabilistic function:

$$w_t(n) \propto \frac{1}{1 - \prod_{(p,n) \in E} w_{t-1}(p) \psi(p, n)}. \quad (6.3)$$

To accommodate this assumption, define ψ as a probability function that maps the expect value between the relation $(p, n) \in E$ such that $\psi(p, n) \rightarrow [0, 1]$. To express $w_t(n)$ probabilistically so that $w_t(n)$ is within the domain of (0,1), the weights of the nodes are calculated thusly:

$$w_t(n) \leftarrow \lambda \left(1 - \prod_{(p,n) \in E} (1 - w_{t-1}(p) \psi(p, n)) \right), \quad (6.4)$$

where $\lambda \rightarrow [0, 1]$ is a path degradation rate and serves a similar purpose as the PageRank damping factor [165], representing the belief that information further from the initial query

is of decreasing relevance. In the case of this research, $\lambda = 0.7$ was used and ψ is:

$$\psi(p, n) = \left| \frac{\log_{10}(\text{expect}(p, n))}{300} \right|, \quad (6.5)$$

the above being an empirical derivation from [104]. All nodes were given initial weights of 0, save the query itself, which is given a weight of 1, and the algorithm iterates until convergence. Applying weights in this manner takes into account the thought that some nodes will be more well-connected than others. Consequently, nodes with more incoming edges will have a higher weight than nodes with less, all other things being equal. The above will be referred henceforth as the *propagation scheme*.

For comparison purposes, another, simpler method of applying weights was additionally used, where each node in the graph is given a weight of 1, *i.e.*:

$$w(n) \leftarrow 1, \text{ for all } n \in V. \quad (6.6)$$

This weighting represents the naïve, high-recall low-precision approach of simply treating all nodes in the graph as equally important in respect to the query sequence. The purpose of this approach is to measure the value of applying weighted metrics to the integrated data, separate from any graph structure; this is termed the *binary scheme*.

Once a graph is weighted, it can be transformed into a feature vector representation. The problem of classification can be generalized within a feature vector space model, as was shown in (3.1) and (3.4). In the case of a query graph, \mathbf{x} can represent a vector of weights from a single data source, with each value therein corresponding to a concrete instance in the query graph. This approach allows one to represent any query graph as several feature vectors, depending on the number of sources, and implicitly captures the presumed relevance of the node under the propagation scheme.

Transformation of the graph weights to fit the feature vector space model is straightforward, and missing data treated simply. Given a data source D with subset of known records H ($H \subseteq D$), the feature vector \mathbf{v} for G on D is:

$$\mathbf{v}^T = \{\text{for all } v \in H : v_0, \dots, v_n\}, \quad (6.7)$$

where,

$$v_n = \begin{cases} w(n) & \text{if } n \in (H \cap V) \\ 0 & \text{otherwise.} \end{cases}$$

In the above, $w(n)$ may take the value of any arbitrary weighting scheme. If \mathbf{v} has known classification, it would then be possible to use it as a member instance of a label in classification training. When a classification method necessitates it, \mathbf{v} may be transformed using some function $\Phi(\mathbf{v})$ to generate a kernel, and thus none of the weighting methods above (propagation or binary) in themselves violate Mercer's condition.

6.1.2 Evaluation methods

The above method and implementation provides a means to query a protein, weight the nodes in the query graph and transform the results into a feature vector representation suitable for training and classification. As the domain of interest is identifying virulent and nonvirulent proteins in bacterial organisms, a curated set of proteins with which to evaluate the performance of our approach was identified in the form of the non-redundant protein set used by Garg and Gupta to test their own virulence detection system [95]. As an added benefit, this allowed for performance comparison of this methodology with previously-published results for the same dataset. The positive, virulent number of examples in the set was 1025, with 820 of these acting as training instances for cross-validation and parameter selection, and the remaining 205 for testing. Likewise, the nonvirulent proteins numbered 1030, with a division of 206 and 824 for testing and cross-validation, respectively. This constituted an 80%-20% train-test split, with the larger fraction used to optimize the parameters for each algorithm and the smaller used for final testing.

The query graphs for the evaluation were generated using the same sources and schema as the evaluation in Ch. 5.3.2 (refer to Fig. 4.5), with the addition of GENNAV [166]. Of the sources incorporated into the schema, all except for ENTREZGENE, ENTREZPROTEIN and UNIPROT were used for classification; the purpose of the omitted sources was primarily to facilitate cross-linkage between other data sources, most notably among them GENNAV and

AMIGO. Though both returned GO terms, the distinction between the two sources was that AMIGO only returned the terms directly being referenced, whereas GENNAV additionally provided the ancestors of terms.

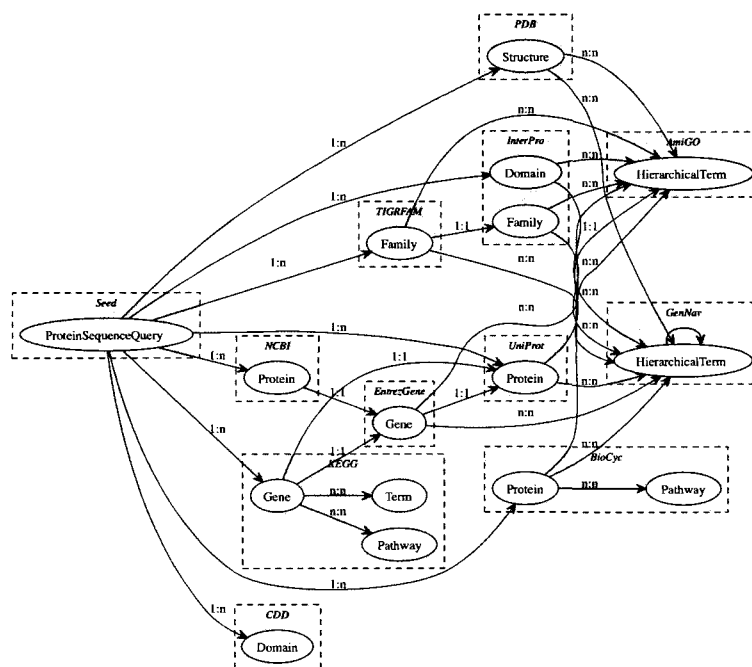


Figure 6.1: Adapted schema used for virulence identification, with CDD, GENNAV and UNIPROT added. Note that GENNAV is a recursive source – that is, it may re-query itself to recreate the GO hierarchy within the query graph.

Query graphs were generated for all 2055 proteins in the evaluation set during the month of October 2008, using the mediated schema (see Fig. 6.1), in MIQAPS. The analysis of the data focuses on two main aspects: characteristics of the structure of the query graphs, and evaluation of performance *via* the AUC, which was chosen because of its usefulness in measuring predictive algorithms and its wide adoption within machine learning research [167]. The AUC is a value between 0 and 1, and is the area under the curve of the true

positive rate against the false positive rate. From a practical standpoint, this value translates into the following: given any random positive instance and any random negative instance, the AUC is the probability that the positive instance will have a higher score than the negative instance, assuming a higher score is indicative of positive instances. Consequently, a classifier whose AUC is 0.5 performs no better than random; conversely, one whose AUC approaches 1.0 denotes perfect classification. As a metric, it has several desirable properties. In contrast to many other measures such as precision, recall and F-score, the AUC is robust to the number of instances in either the positive or negative set [168]; in comparison to accuracy, the AUC is more informative and the resulting curve can be used as a guide to choosing an ideal threshold. Efficient calculation of the AUC for any given classifier, while straightforward, involves several steps, and one may refer to [167] for further details.

Three learning algorithms were tested to evaluate whether the approach can be robustly applied to different classifiers: SVMs, ridge regression and k nearest-neighbor (k NN) [116, 169, 170]. Publicly-available implementations³ of these algorithms were used in conjunction with the query graphs. Because these methods have varying degrees of tolerance to noise, this would also allow the level of spurious information in the data sources to be gauged by each algorithm's relative performance. The parameters optimized were the following for each algorithm: regularization cost C and kernel (linear, Gaussian, polynomial) function (with appropriate width for the Gaussian⁴ and power for polynomial) in SVMs; ridge parameter in ridge regression; and neighborhood size in k NN. The train-test process was applied for each source, in order to determine their individual predictive ability, and the final test results compared.

6.2 Results

6.2.1 Query results characteristics

The retrieval of abundant data relevant to a query protein is one of the lynchpins of this research. However, many proteins lack significant sequence similarity to sequences from

³PyML, <http://pyml.sourceforge.net>

⁴The Gaussian kernel is defined as: $k(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}}$, and the polynomial as $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^z$, where σ is the width and z is the power for the Gaussian and polynomial, respectively.

outside their genera and a notable fraction of these proteomes are “hypothetical”. For example, approximately 12% of *Shigella* proteins are hypothetical, and for *Pseudomonas* this number approaches 30% [171, 172]. Using a data integrative approach, however, yielded sufficient numbers of returned records for at least one data source. This was particularly true for GENNAV and AMIGO, which provided results for more than 90% of the query proteins (see Tbl. 6.1). Recall that these sources rely primarily on cross-linkage from other sources, and so the GO terms in the graph are not associated with the query sequence directly, but rather may be transitively connected through similar sequences.

<i>Data src.</i>	<i>Fraction of coverage</i>		<i>Source size</i> (in appr. units)
	<i>Virulent</i> (<i>n</i> = 1025)	<i>Non-virulent</i> (<i>n</i> = 1030)	
AMIGO	0.91	0.99	2500 terms
BIOCYC	0.39	0.60	1347 proteins, pathways
CDD	0.72	0.93	4640 models
GENNAV	0.91	0.98	3355 terms
INTERPRO	0.86	0.96	2256 models
KEGG	0.41	0.53	150 pathways
PDB	0.48	0.72	4443 molecules
TIGRFAM	0.38	0.67	1055 models

Table 6.1: Database coverage by fraction across all sources for the training and test sets by MIQAPS. Source sizes are estimates as of January 2009.

Consistent across all data sources, there was more information available for non-virulent proteins than for virulent proteins. The difference is particularly stark for TIGRFAM, which has a broad focus of coverage across the microbial proteome [149]. Similarly, PDB and BIOCYC have significant differences in coverage between virulent and non-virulent proteins. One possible explanation is that a large number of non-virulent proteins are involved in metabolic activities, which are likely to be conserved across bacteria. While many virulence factors across pathogens play similar roles and may share homology, the mechanisms themselves may vary [173]; this is supported by the coverage differential between KEGG and BIOCYC. While both are pathway-centered databases, BIOCYC is more heavily influenced

by well-curated metabolic data, and KEGG includes more organisms in its database – 919, the vast majority of which are bacterial, to BIOCYC’s 371.

An examination of the query graphs themselves revealed a non-normal skew towards a high number of poorly-connected nodes, and a comparatively small number of well-connected nodes. The query graphs are not strictly scale-free, though various sections of the cumulative distribution may be interpreted to be power-law derived based on the results of a nonlinear fitting to the probability distribution (see App. B.1), which complements the findings of the fault tolerance experiment in Ch. 5.4.

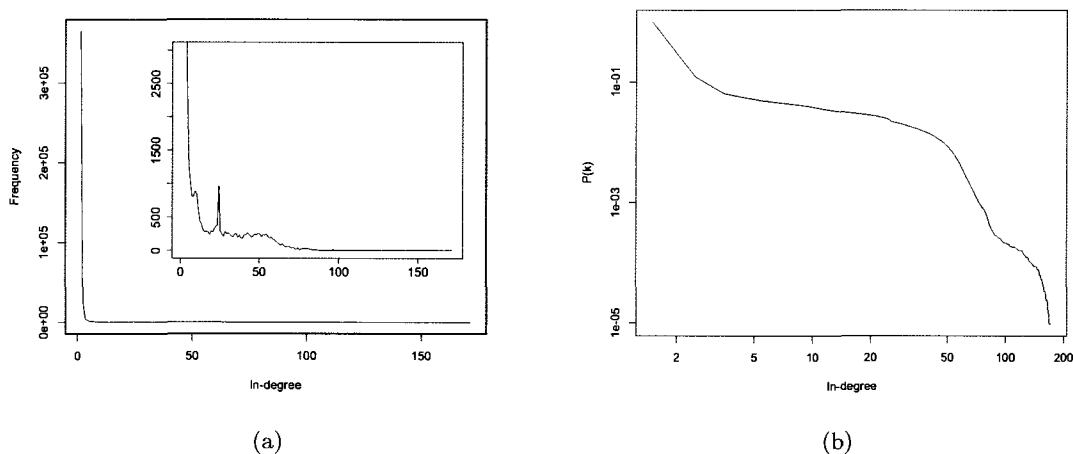


Figure 6.2: In the above, Fig. 6.2a is the frequency of the in-degrees of nodes in all 2055 query graphs (with detailed inset), and Fig. 6.2b is a log-log plot of the in-degree cumulative probability distribution; the seed nodes are omitted in these charts.

Weight frequencies of the propagation scheme displayed similarly atypical distributions (see Fig. 6.3), ones which appear to reflect the expansion-based structure of the graph. Data source entities that included intra-links to other entities or that were linked indirectly to the query node exhibited multimodal frequencies with a periodicity equivalent to powers of the path degradation rate λ (see Fig. 6.3 for frequencies). This is most apparent in GENNAV, where the numerous frequency spikes beyond those at the ends reflect a traversal

of the hierarchy. Furthermore, data sources that search against shorter sequences or motifs, such as CDD and TIGRFAM, have relatively fewer nodes that score on the higher end of the weighting spectrum – likely as a result of the expect value conversion used (6.5). All of these could have effects on the use of any given data source as a discriminator between virulent and non-virulent protein classes.

6.2.2 Performance across sources, learning methods

The performance of using data sources individually as a means of predicting relation to virulence was measured *via* the AUC. For comparison with VirulentPred, from which the training and testing set was derived, accuracy was also used as a performance metric (see Tbl. 6.4).

<i>Data source</i>	<i>Classification method</i>		
	SVM (RBF)	Ridge regr.	kNN
AMIGO	0.894	0.907	0.867
BIOCYC	0.698	0.687	0.679
CDD	0.729	0.760	0.755
GENNAV	0.940	0.935	0.878
INTERPRO	0.846	0.804	0.832
KEGG	0.733	0.778	0.779
KEGG (pathways)	0.740	0.739	0.717
PDB	0.740	0.737	0.710
TIGRFAM	0.688	0.702	0.704

Table 6.2: Results by source and method for predicting virulent and non-virulent bacterial proteins given AUC. The best performer, GENNAV was run with a Gaussian kernel whose $\sigma = 1.0$ and regularization cost $C = 1.0$.

One emergent pattern from the results was that the more coverage a data source provided, the better it performed. The notable exception is the difference between AMIGO and GENNAV – both sources use GO terms linked from other sources, and have the same coverage. However, GENNAV links to the parents of the GO terms, and the parents of those GO terms and so on, up to the top-level of the GO hierarchy. Despite the similarity in cov-

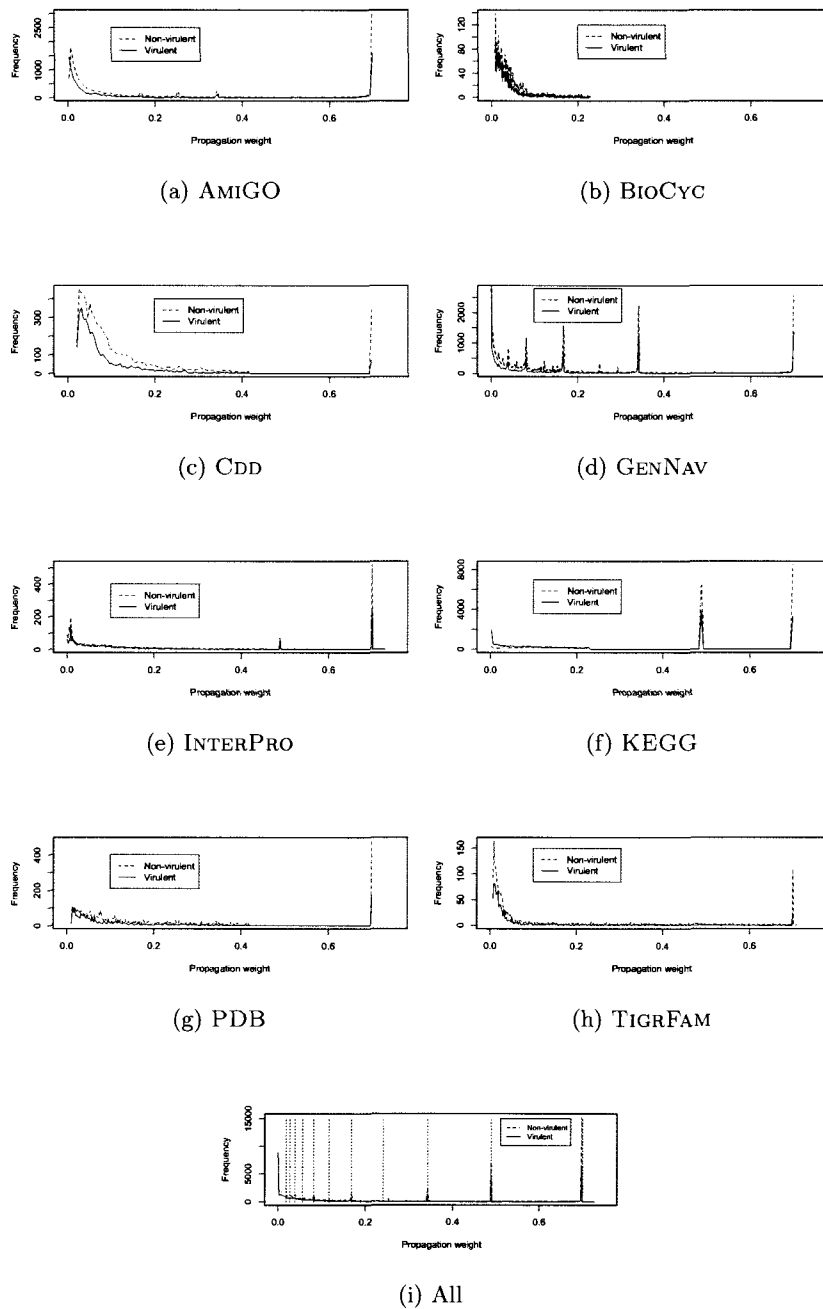


Figure 6.3: Frequency plots of the propagation weights across non-virulent and virulent proteins, by source; Fig. 6.3i shows the periodic influence of λ on discretizing records beyond the query node.

erage, GENNAV outperforms AMIGO by as much as 0.05. GENNAV generates more data than AMIGO *via* self-reference, and the performance difference suggests that leveraging the ancestry of a GO term may be more useful for predictive purposes than just the immediate GO term by itself. Overall, results imply that the sources oriented around GO terms were the best performing, while TIGRFAM and BIOCYC were the least predictive.

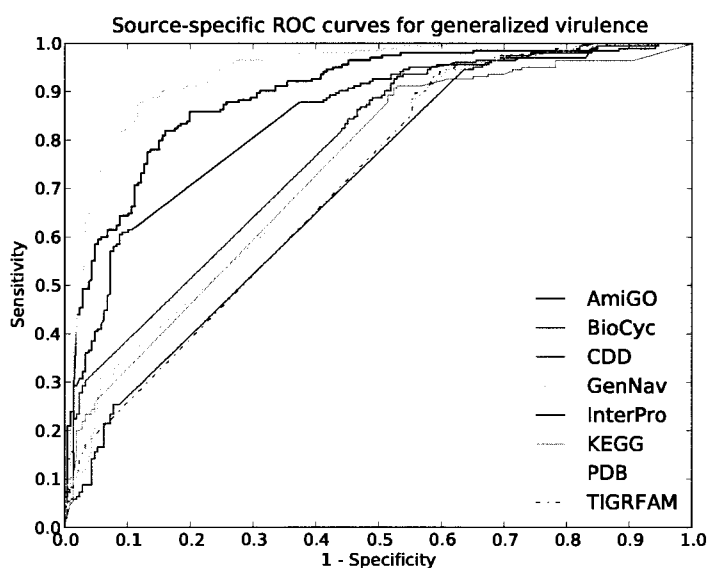


Figure 6.4: ROC curves for the eight different data sources based on the test data. Parameters were coarsely optimized for the AUC.

To see if the pattern carried over when empty query graphs were excluded, the same train-test process was re-ran as before, omitting any query graph from the training or testing that did not yield any query results. As the SVM approach seemed to do the best on average compared to ridge regression and k NN, that statistical learning approach was used for the re-run, and the appropriate parameters were optimized for this subset of the training-testing data as described in Ch. 6.1.2.

Omitting empty graphs reduced the number of training and testing instances for each source, in some cases by more than 50%. However, the result was a rough sense of the

predictive ability of each source, given records existed for that source in the query graph (see Tbl. 6.3). Though AMIGO and GENNAV maintained essentially the same scores, the rest of the sources experienced noticeable increases. Despite this overall improvement, the relative ranking of the sources remained the same, again with AMIGO and GENNAV outperforming other sources.

<i>Data source</i>	<i>AUC</i>
AMIGO	0.886
BIOCYC	0.807
CDD	0.876
GENNAV	0.940
INTERPRO	0.883
KEGG	0.795
KEGG (pathways)	0.815
PDB	0.875
TIGRFAM	0.872

Table 6.3: Above are results by source, when empty graphs (query graphs with no returned results) are excluded from training and testing; the scores are thus those of each source given data from that source was available.

Comparing the AUCs and accuracies of using weighted and integrated queries with the cascaded SVM approach, there is a marked improvement in performance. Using the best-scoring single source (GENNAV), the three learning approaches were compared to a sequence baseline and *VirulentPred*, the cascaded SVM system developed by Garg and Gupta (see Tbl. 6.4); this comparison uses the first train-test phase that includes all query graphs (including those with empty query results). Regardless of the statistical learning method used, GENNAV integrated queries resulted in AUCs of 0.07-0.08 higher than the cascaded SVM approach, and approximately 0.15 greater than the sequence baseline. Accuracies are less one-sided, and in fact the k NN approach did only 0.053 better than the sequence baseline, suggestive of the significant amount of noise present in the retrieved data.

<i>Data src.</i>	<i>Class. method</i>	AUC	Accuracy
1-mer	SVM (RBF)	0.786	0.710
-	VirulentPred	0.860	0.818
GENNAV	SVM (RBF)	0.940	0.868
GENNAV	Ridge regression	0.935	0.863
GENNAV	kNN	0.935	0.763

Table 6.4: Comparison of the top-performing integrated predictor against a sequence baseline and **VirulentPred**; all methods outlined in the table used the same set of proteins.

6.2.3 Comparison of query graph weighting schemes

The results of the previous section rely on the more complex propagation scheme, which attempts to take into account the structure of the graph to elevate query results that are well-referenced by assigning them relatively higher weights under the assumption that those results better represent the query. In the binary scheme, the primary interest is in whether or not there is value in the abundance of data itself – if the fact that the data was retrieved at all can be predictive, independent of the path taken from the query to retrieve it.

Fig. 6.5 compares node-weighting using the propagation scheme with node-weighting with the binary scheme, both *via* the SVM learning method. As it was for propagation data, cross-validation and training-testing for the binary scheme was conducted to determine optimum parameters and final scores. Other than INTERPRO, BIOCYC and PDB, which seems to perform better with propagation weighting, most data sources perform nearly equivalently using either the propagation or binary scheme. Differences between binary and propagation AUCs were most visible for less predictive sources, but for the highest-scoring data sources the differences are negligible.

6.3 Conclusion

Based on these findings, one can draw several conclusions on the use of fractured databases as sources of predictive information. Surprisingly, it was found that the data sources several nodes away from our query were the most predictive, which may at first glance seem counter-

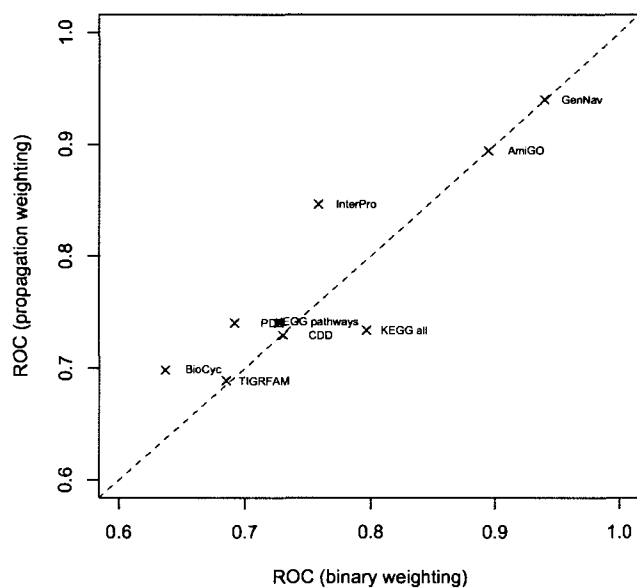


Figure 6.5: Plot of ROC scores between using propagation-weighted nodes (y -axis) and binary-weighted nodes (x -axis); done using SVM-based classification.

intuitive. GENNAV and AMIGO both are source indirectly connected to the initial queries in the query graph, yet they were the best performing. A large part of this may be due to their superior coverage, as they are referenced by other data sources, but even when empty graphs are omitted it was found that these data sources still out-scored the other sources directly connected to the query. This reinforces the position that, as it pertains to using biological databases for classifying protein data, shallow queries are generally not sufficient. Manually, this information can be difficult to sift through, and using robust methods (*e.g.*, statistical learning) to cut through the “chaff” is indispensable. This is particularly salient given the results that learning methods more resistant to noise (SVMs, ridge regression) were the best at identifying virulent proteins. At the same time, latent information in many databases are redundant to each other, as the results with using an unweighted integrated kernel seems to suggest. This is in corroboration with earlier findings in Ch. 5 regarding

fault tolerance.

A limitation which is not explored in this chapter but is self-evident in implementation is that the performance will likely be very dependent on the choice of data sources and cross-linkages. Nonetheless, one can find here that there is value in cross-linking across data sources, and that while provenance and quality of data is naturally very important even the most naïve retrieval approaches can provide useful information on identifying protein virulence, and perhaps protein class in general. A possible conclusion is that data which is most indicative of characterizing a protein and that which is less indicative may not necessarily reflect a continuous distribution of relevance. This is supported by the relatively equivalent performances between the propagation and binary schemes for many sources. Interestingly, though, it appears as if the propagation method works best for sources closest to the originating query, which either suggests that there is some significant information loss as sources further from the query are explored, or that some sources are simply more informative in themselves than others for virulence classification, and that propagation scoring does not add any further significant value to predictive ability.

The methods outlined in this chapter were applied to a very broad class of proteins, and it is the assertion of this dissertation that the process described is readily generalizable to protein classifications of other types as well, since no single step other than the choice of training and testing set is virulence-specific. Furthermore, as the foundation of the method is built on the MIQAPS data integration framework and not a specific model of integration, extending and evaluating the work beyond the sources and cross-links outlined here would be as simple as providing interfaces to the sources and defining the inter-source connections.

While the data sources may be different, generally graphs of cross-linked biologic databases tend to exhibit similar structure. Redundancy of data suggests there is some considerable robustness in these graphs concerning interconnectedness, again supporting the findings from Ch. 5. Moreover, from a biological standpoint, these conclusions validate the position that information for virulence proteins is less available in biological data sources than that of the general protein population. In a feedback loop, this also makes it more difficult to elucidate mechanisms of infectious disease pathogenicity using traditional, shallow bioinformatics methods, and seems to emphasize the importance of continued work in providing

integrated access to virulence-related data.

6.4 Discussion

The benefits of combining classification methods and data integration include the synergistic and complementary nature of the two fields. First, as alluded to in Ch. 2, wet-lab experiments are costly, and without proper prioritization assays can be fruitless when a specific goal is in mind. Computational methods, while inexpensive, often yield high-quality information only for well-known proteomes at best, and speculative and unverified information for others. Data integration provides a means of generating sufficient functional coverage for data using non-experimental query-based methods, while statistical learning techniques such as SVMs provide a means of deriving information from the queried data of tunable quality. By using path-based integration and exploratory queries, coverage is greatly increased, albeit at the cost of likely greater noise.

This chapter outlined a method of combining statistical learning and federated data integration methods by weighting query results into feature values *via* an iterative process, and then transforming those weights into learnable representations of query graphs. Thus, protein queries (and theoretically, queries of any arbitrary type) become examples that can be used for classification by leveraging the mapping space of available public biologic data sources. This approach was tested on a known dataset of virulence proteins, with the finding that it outperforms a previously-developed approach using similar classification methods, but without data collected by integrated queries.

While being able to identify generalized virulence proteins is of some use, specific roles in virulence would be of greater utility to a biomedical scientist. Imagine, for example, a biologist whose interest is specifically in proteins likely involved in biofilm creation – an extracellular matrix generated by colonies of bacteria used for *in vivo* survival. Being able to identify with high precision proteins involved in biofilm development would be more informative in this case than merely virulence in general. The next chapter addresses this issue, and generalizes the methods in this chapter by expanding integrated queries to learning specific virulence roles.

Chapter 7

LEARNING PROTEIN INVOLVEMENT IN SPECIFIC PATHOGENIC ROLES

The results presented in Ch. 6 strongly suggested that a learnable query graph can be used for specific virulence prediction. Indeed, the performance of using a subset of nodes (GO terms) from a query graph outperformed previously-attempted methods at generalized virulence prediction. While these findings are encouraging, prediction of virulent proteins is of limited value without specific identification of the role or function the protein may play in pathogenesis and infection maintenance. A classifier that could provide a researcher with not only the information that a protein is involved in virulence, but also in what capacity, would be more useful for gene prioritization. To that end, this chapter discusses the methods and results associated with assigning specific roles of virulence for pathogenic proteins.

7.1 Methods

7.1.1 Dataset preparation

While the dataset developed by Garg and Gupta and used in the general virulence prediction experiment in Ch. 6 was curated, it lacked the annotation granularity needed to determine particular virulence roles a protein may play; the dataset was purely binary in classification, and a protein was categorized as either ‘virulent’ or ‘non-virulent’. For a more specific prediction of virulence factors, a new, and ostensibly larger, dataset was necessary that contained not only information on whether or not a protein was involved in pathogenesis but also conveyed information regarding in what capacity. In Ch. 2.1.1, a number of pathogen-specific databases were listed, among them MVIRDB. As a data warehouse for pathogenic proteins, MVIRDB provided a rich resource for a number of proteins that were involved in infectious disease, including information that was confirmed through literary

review and experimental results [27]. Because of its comprehensiveness MVIRDB was used as the primary resource for generating a dataset with finer-grained classes of virulence for the experiments described later in this chapter.

In order to transform the protein data in MVIRDB into a suitable training and testing set, the first step was curation of the data into a non-redundant, representative set of proteins. The original MVIRDB dataset used consisted of 14544 records, primarily of amino acid sequences with some genomic sequences interspersed throughout. DNA sequences were translated to protein sequences, beginning at the methionine if present and using the longest open reading frame; otherwise, the DNA sequence was removed from the set. Databases whose contents were viral sequences were removed from the set; these initial filters yielded 5052 remaining proteins. For negative instances, 3000 proteins were randomly drawn from GENBANK [151] and filtered for proteins highly likely to be involved in virulence based on regular expression searches on the protein names and annotations. For example, proteins whose names contained ‘drug’ or ‘toxin’ were removed. Proteins from known pathogenic organisms were otherwise left undisturbed in the negative set, as presumably not all proteins within an infectious organism are involved in virulence. At the same time, hypothetical proteins whose functions were unknown were also removed from the negative set. Finally, CD-HIT [174] was used to generate non-redundant protein clusters at a 40% level of identity. The final sequence dataset consisted of 3700 proteins, 1703 of which constituted the negative (non-virulent) set and 1997 of which formed the positive (virulent) classes.

7.1.2 Curating virulence factor subclasses

Once the datasets were curated for non-redundancy (in the positive set) and possible virulence factors (in the negative set), the positive set proteins were classified into various virulence functions. Classifications were done based on the information regarding the protein readily available from the virulence data sources; many of the databases that MVIRDB integrated used a native classification system. For example, one data source included in MVIRDB was VFDB [23], a database with a deep hierarchical classification system for virulence factors consisting of four top-level terms (‘Offensive virulence factors’, ‘Defensive

virulence factors’, ‘Nonspecific virulence factors’ and ‘Regulation of virulence-associated genes’) and 36 lower-level terms ranging from ‘Adherence’ to ‘Type VII secretion’ to ‘Toxin: intracellular toxin: DnaseI’. Though MVIRDB provided uniform access to the data within these other sources, methods of classifying and organizing virulence factors were not aligned in any way, and thus the proteins within MVIRDB were classified *via* a number of different, and often quasi-overlapping, umbrella terms. The challenge of aligning the terminology and classifications of virulence factors is well-known, and has been noted as a major impediment to consolidation of virulence data [175].

Because of the misalignment of classifications in the dataset, virulence proteins were annotated manually by the author of this manuscript, based on the original classifications and literature references of the native databases. To illustrate the need for manual annotation over the positive dataset, many databases whose focus was on a specific type or family of proteins, such as in the case of ARGO and antibiotic resistance proteins, simply annotated all proteins as a single type. As a result, a small number of categories have very many instances. In other cases, annotations appeared idiosyncratic at the deepest level, but may have been subsumed by higher-level annotations. In this regard, the problem faced is similar to that encountered by the curators of the Unified Medical Language System (UMLS), the Foundational Model of Anatomy (FMA) and GO [176, 177, 102] and similarly a solution based on manual alignment of the various databases’ classifications schemes is used here. While automated and semi-automated methods exist for the ‘ontology alignment’ problem (*e.g.*, see [178, 179]), manual methods were used for the positive dataset to ensure the highest quality annotations possible, given the limitations to manpower and time. This manual annotation process is outlined stepwise in Tbl. 7.1.

Per the steps, manual annotation of the virulence proteins was an iterative process that continued until no further classification changes were made to the dataset (either added, changed or deleted). For the positive dataset, three iterations were done before changes were no longer made and the classifications were deemed to have reached a steady state. As a result of the manual annotation, 28 different hierarchical virulence-related classifications were derived. Because many of these classifications contained a small number of proteins unsuitable for instance-based classification, only the 11 top-level classes were

<i>Procedure for manual curation of virulence factors</i>
<ol style="list-style-type: none">1. Examine the source or database of each protein annotation for possible classifications, using the annotation set across all databases as a starting point. Record annotations according to information from the source or database; each protein may have more than one annotation. If a protein is directly involved in a virulence process or <i>is a regulator</i> of that process, record it as such. In this way, proteins may have more than one annotation.2. Examine any publications which are <i>linked from the source</i>. Record annotations according to information from the publication regarding the protein.3. If an annotation was unclear or unknown, conduct a <i>keyword publication search</i> of the virulence factor to obtain resolution.4. Repeat steps (1-3) across all proteins (<i>i.e.</i>, re-annotate) until no further changes were made from the previous annotation.

Table 7.1: Iterative method used to manually align and annotate the virulence classifications for pathogenic proteins in the training and testing dataset.

used for virulence prediction (see Tbl. 7.2). There were a small number of proteins whose status as virulence factors could not be confirmed, or whose evidence appeared suspect upon review of annotation or literature; these proteins were removed from the dataset. Across all levels of the curated virulence classification hierarchy, the annotation of each protein to a coarse-level virulence annotation was not done *de novo*; annotations were based on the annotation and classification information for each protein entry in the originating databases. Moreover, in cases where further investigation of a protein was required for clarification, only literary evidence was used; sequence searches were omitted from the annotation process, which would have directly influenced the results of the automated classification.

7.1.3 Baseline classifiers

Unlike the dataset used in Ch. 6.1, the dataset whose preparation was outlined in Ch. 7.1.1 is an artifact created for the purposes of exploring the methods described in this dissertation, and thus there are no existing approaches upon which it has been tested. Subsequently, three classifiers were developed as baseline systems for comparisons to query-level integration and learning.

The first such baseline approach used *3mer* sequence frequencies, as was employed for comparison in Ch. 6.1. The use of a purely sequence-based classifier again provides an *ab initio* baseline for the classifier, and represents a simple but effective remote homology detection algorithm [93]. The feature space for this classifier was calculated on-the-fly for each protein sequence, and indexed by the space of all *3mer* words seen in the training dataset.

A BLAST database of the specific virulence dataset was created, and the second baseline classifier was based on the mutual BLAST results of the dataset proteins against each other. Each individual protein i was queried against the created BLAST database, and its affinity p to any given class $L \in \mathcal{L}$ was determined by:

$$p_{i \in L} = \frac{\sum_{n \in N_i^{(k)}} \mathbf{1}_n(L)}{|N_i^{(k)}|}, \quad (7.1)$$

where the set $N_i^{(k)}$ denotes the neighborhood of k -nearest proteins to i (as determined by

Virulence Categories

(1)	Adherence (<i>e.g.</i> , pili, fimbriae)						
(2)	Surface factor (<i>i.e.</i> , outer membrane proteins, porins)						
(3)	Invasion-related <table style="width: 95%; border-collapse: collapse; margin-left: 20px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Entry (into host cell)</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Spreading factor (facilitation of pathogen spread or population growth, <i>e.g.</i>, biofilm)</td> </tr> </table>	Entry (into host cell)	Spreading factor (facilitation of pathogen spread or population growth, <i>e.g.</i> , biofilm)				
Entry (into host cell)							
Spreading factor (facilitation of pathogen spread or population growth, <i>e.g.</i> , biofilm)							
(4)	Transport and uptake (metal and nutrient transport related to pathogenesis) <table style="width: 95%; border-collapse: collapse; margin-left: 20px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Iron uptake and storage</td> </tr> </table>	Iron uptake and storage					
Iron uptake and storage							
(5)	Toxin (evocation of host tissue damage or cytotoxicity) <table style="width: 95%; border-collapse: collapse; margin-left: 20px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Exotoxin</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Endotoxin</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Bacteriocin (toxin specific to host bacterial flora)</td> </tr> </table>	Exotoxin	Endotoxin	Bacteriocin (toxin specific to host bacterial flora)			
Exotoxin							
Endotoxin							
Bacteriocin (toxin specific to host bacterial flora)							
(6)	Catalysis (<i>e.g.</i> , protease, proteinase)						
(7)	Secretion <table style="width: 95%; border-collapse: collapse; margin-left: 20px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Type I</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Type II</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Type III</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Type IV</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Type V</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Type VII</td> </tr> </table>	Type I	Type II	Type III	Type IV	Type V	Type VII
Type I							
Type II							
Type III							
Type IV							
Type V							
Type VII							
(8)	Motility						
(9)	Antibiotic resistance						
(10)	Resistance and defense (<i>e.g.</i> , intra-host survival, stress adaptation, anti-immune effects) <table style="width: 95%; border-collapse: collapse; margin-left: 20px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Antiphagocytosis</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Immune evasion</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Antigen and antigenic variation</td> </tr> </table>	Antiphagocytosis	Immune evasion	Antigen and antigenic variation			
Antiphagocytosis							
Immune evasion							
Antigen and antigenic variation							
(11)	Other <table style="width: 95%; border-collapse: collapse; margin-left: 20px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Cell wall</td> </tr> <tr> <td style="border-left: 1px solid black; padding-left: 5px;">Transduction</td> </tr> </table>	Cell wall	Transduction				
Cell wall							
Transduction							

Table 7.2: 11 top-level virulence factor categories derived from the positive training and testing set, with subclassifications. Importantly, a virulence factor may be classified under several labels, *e.g.*, a protein may be both a ‘Surface factor’ (label 2) and an ‘Antigen’ (label 10).

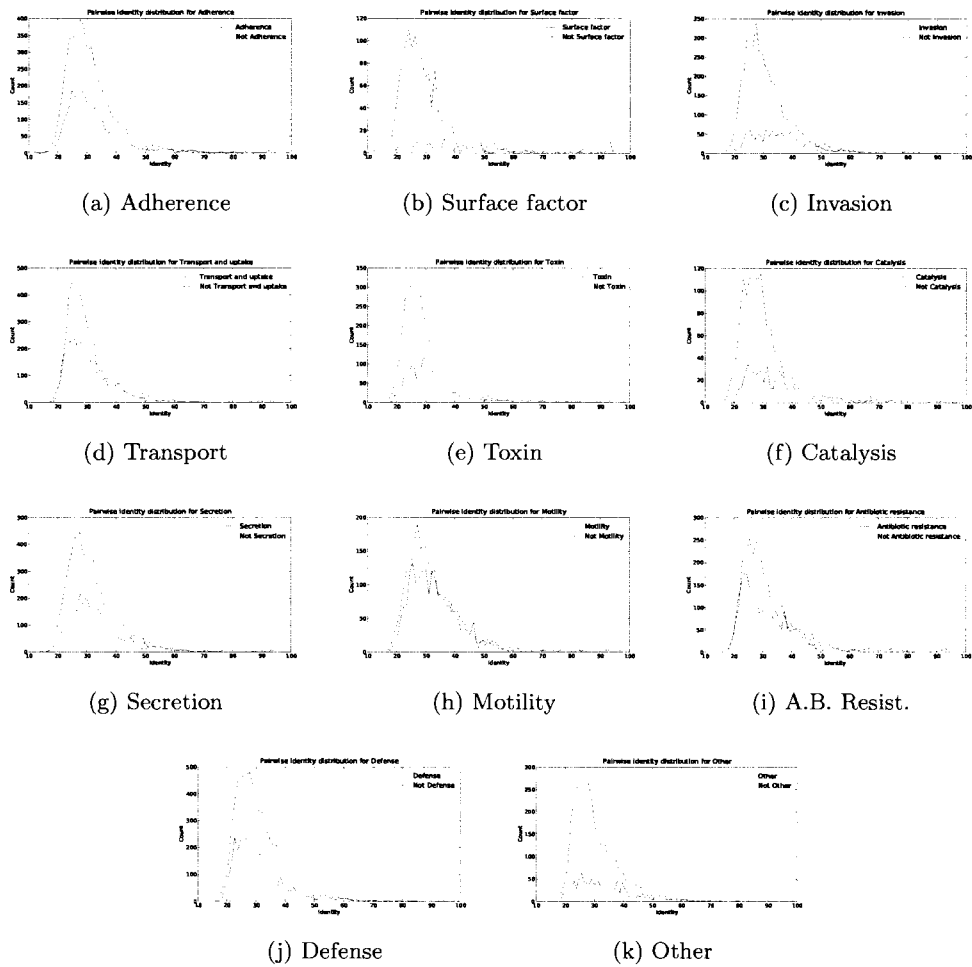


Figure 7.1: Inter- and intra-category similarity distribution by identity, pre-40% identity clustering and using blastall with the low-complexity filter off

highest results from BLAST), and $\mathbf{1}_n(L)$ is the indicator function, which is equal to 1 if $n \in L$ and 0 otherwise; thus, p is the fraction of the k -nearest neighbors of i that have membership in L . This approach was used since each protein in the dataset could take on multiple classes at once and the formulation in (7.1) permits the measurement of membership strength for any arbitrary class, given some protein. For the purposes of this dissertation, the cluster size was chosen to be $k = 3$, based on its popular usage in previously-published experiments using nearest-neighbor methods on protein sequences. The motivation behind this very simple approach is to measure annotation based on data from a single source, and in such a way as to emulate how an annotator may scan the best-scoring BLAST hits of a sequence to determine function [19].

The third and final baseline classifier used is also based on BLAST, but relies on a more sophisticated approach – indeed, it uses SVMs trained on pairwise hits (with a high e-value threshold) against a BLAST database of the training set. To generate features for this third classifier, each test sequence was queried against the trained BLAST database, resulting in a vector representation of a sequence’s negative log transformed e-value score to the other sequences within the database. This method is referred to as BLAST+SVM and a similar method has been used in past research, where SVMs based on pairwise BLAST queries outperformed or were comparable to other methods such as SVM-Fisher, SAM, PSI-BLAST, Smith-Waterman and motifs+SVMs in detecting sequences that were remotely homologous [180, 181],

7.1.4 *Evaluation methods*

Unlike the prediction of generalized virulence in the previous chapter, the problem of specific virulence as it has been presented here is a multiclass problem. Each protein is permitted to have multiple virulence labels attached to it, and thus for each classification method and source 11 different SVMs were tested in a one-versus-rest fashion. That is, each virulence category was set as the ‘positive’ set of interest, and all other proteins (nonvirulent proteins and virulent proteins of a differing classification) were treated as the ‘negative’ set. Two primary experiments were conducted on the specific virulence set.

First, similarly to how evaluation was conducted for generalized virulence, the dataset for specific virulence was split into a training and testing components, and 80% of the dataset was used for training the various parameters for the classifiers and 20% retained for final testing (refer to Tbl. 7.3 for details). For the integrated query graphs, data was generated as in the generalized virulence experiment, with the same data sources and identical schema. As the Gaussian SVM performed the best overall in the generalized virulence experiment, this kernel was chosen for all SVMs in the specific virulence classification experiment. Optimal parameters for the integrated query graph were determined *via* a grid search on the width and cost of a Gaussian kernel, again using the same procedure as generalized virulence. The parameters selected for each source and for each virulence class were those that provided the best AUC performance.

One step utilized in specific virulence that was not done in general virulence was *feature selection* on the integrated query graphs for the train-test split experiment, in the form of *F-scores*. This metric, calculated prior to SVM training and testing, provides a rough estimate of the predictive value of a feature, independent of the other features, for any given class. The addition of this step in the evaluation process seemed appropriate, given the larger size of the dataset, and thus the larger number of features expected to be returned per data source (the details of source coverage are in Ch. 7.2). The calculation used here follows the formulation outlined in [182]. Let i correspond to the i^{th} feature in a data source. Then:

$$F(i) = \frac{(\bar{x}_i^{(+)} - \bar{x}_i)^2 + (\bar{x}_i^{(-)} - \bar{x}_i)^2}{\frac{1}{|\mathbf{x}^{(+)}| - 1} \sum_{k=1}^{|\mathbf{x}^{(+)}|} (\mathbf{x}_{k,i}^{(+)} - \bar{x}_i^{(+)}) + \frac{1}{|\mathbf{x}^{(-)}| - 1} \sum_{k=1}^{|\mathbf{x}^{(-)}|} (\mathbf{x}_{k,i}^{(-)} - \bar{x}_i^{(-)})}, \quad (7.2)$$

where, respectively, $\mathbf{x}^{(+)}$ and $\mathbf{x}^{(-)}$ are the positive and negative datasets, $\bar{x}^{(+)}$, $\bar{x}^{(-)}$, \bar{x}_i are the averages of the positive, negative and complete sets of the i^{th} feature, and $\mathbf{x}_{k,i}^{(+)}$, $\mathbf{x}_{k,i}^{(-)}$ are the values of the i^{th} feature of the k^{th} instance of the positive and negative sets. Features whose F-scores were in the top 25%, 50% and 75% were tested, per source in the training set. As with the SVM parameters, the features that yielded the best AUC in training were the features then used for the final test results.

The second experiment involved running six five-fold cross-validations for each class and

<i>No.</i>	<i>Virulence category</i>	<i>Instance count</i>
1	Adherence	360
2	Surface factor	66
3	Invasion	249
4	Transport and uptake	225
5	Toxin	319
6	Catalysis	84
7	Secretion	483
8	Motility	181
9	Antibiotic resistance	239
10	Defense	488
11	Other	214

Table 7.3: The 11 main virulence categories derived manually from the pathogenic protein data sources with the number of training and testing records, after 40% identity pruning.

method with the intention of obtaining measures of variance and deviation for each classifier. For each cross-validation run, the five-fold splits were the same across all classifiers to accommodate direct, paired comparison. In the case of SVM-based baseline methods and sources, Gaussian kernels were used, as they seemed to have the most consistent performance in generalized virulence prediction; parameters for the kernel and SVM were default and non-optimized, per LIBSVM [183]. Because 30 individual values are reported for each classifier per virulence class, paired two-tailed t -tests were used to measure the significance of any mean differences between the sources, and between the sources and baseline methods. Furthermore, because of the statistical power provided by 30 paired values, all integrated sources and baseline methods were used for this experiment and p -values were conservatively adjusted for multiple pairwise comparisons *via* Bonferroni correction.

7.2 Results

7.2.1 Coverage and summary statistics

For virulence proteins, coverage using data retrieved *via* formal data integration methods were similar to those yielded by the Garg and Gupta dataset (see Tbl. 7.4; refer back to

Tbl. 6.1 for generalized comparison). However, the coverage for nonvirulent proteins was considerably different, with the randomly-selected negative dataset for specific virulence being slightly less characterized than the negative set used for generalized virulence testing. Furthermore, in the specific nonvirulent case the level of characterization is less than that of the virulent proteins – the opposite of what was found in the generalized virulence experiments. This discordance is likely due to the choice of databases and annotation level from which the negative instances were drawn. The negative set used by Garg and Gupta were taken from the SWISS-PROT database [152], a database of manually curated and annotated proteins. One would expect, then, that these proteins would be fairly well characterized, and given the standing the SWISS-PROT database has within the molecular biology community it would be of little surprise if a number of these annotations from SWISS-PROT propagated to other databases as well.

<i>Data src.</i>	<i>Fraction of coverage</i>		
	<i>Virulent</i> (<i>n</i> = 1997)	<i>Non-virulent</i> (<i>n</i> = 1703)	<i>Num. features</i> (in appr. units)
AMIGO	0.91	0.80	5102 terms
BIOCYC	0.43	0.37	1674 proteins, pathways
CDD	0.72	0.70	6463 models
GENNAV	0.90	0.83	6425 terms
INTERPRO	0.86	0.87	3540 models
KEGG	0.42	0.57	234 pathways
PDB	0.49	0.55	7954 structures
TIGRFAM	0.39	0.29	1109 models

Table 7.4: Database coverage by fraction across all sources for the training and test sets by MIQAPS for the specific virulence dataset.

On the other hand, the negative dataset used for specific virulence prediction was drawn from GENBANK, and is expectedly less well-characterized. Arguably, this dataset is a more accurate reflection of available biologic information, since the random selection resulted in a mix of proteins of varying levels of annotation. Intuitively, the coverage for the specific dataset also match what one would expect from an automated annotation system dealing

with live data, and trends with the 70% ‘hurdle’ of easily-annotatable proteins [184]. Given the larger size of the specific virulence dataset (almost double that of the general virulence dataset), a larger feature space was present. This was expected, and the F-score selection process (Ch. 7.1.4) was included partly to lessen the computational resources the higher feature space would require in training and testing.

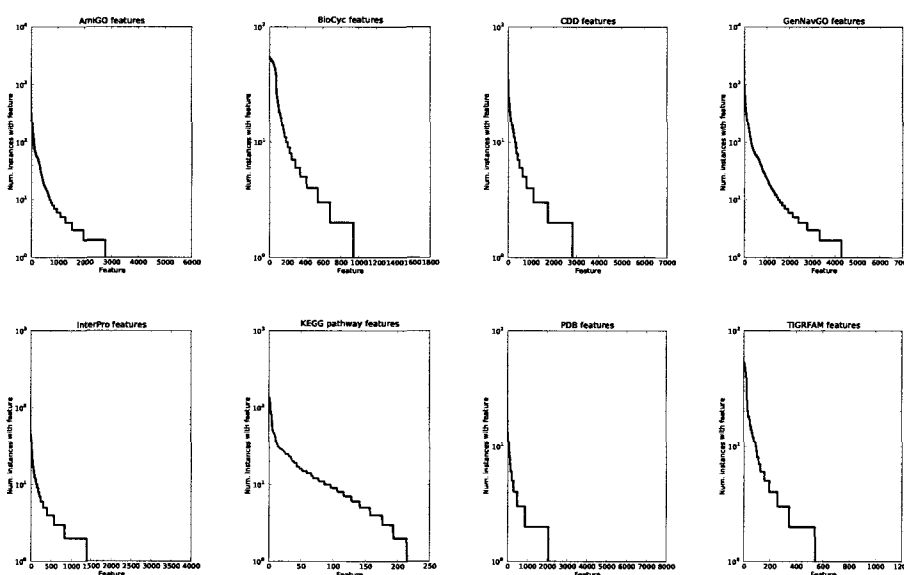


Figure 7.2: Feature distributions by source on log-y plots; the x-axis refers to individual features, sorted descending by the number of instances in which they appear (in the y-axis). Less precipitous drops indicate sources where the appearance of any given feature in any instance is more evenly distributed.

One argument for using integrated queries across multiple data sources is to provide adequate and informative coverage of results for a query that could not be met by a single source on its own. As was the case in generalized virulence, several data sources generated very sparse matrices. Recall that the primary method used for retrieving results directly related to the query was through sequence comparisons to individual databases, and many proteins in the dataset did not have any homologous proteins in some sources. Subsequently,

for many sources, a feature will appear in only one instance, and never again, though some features appear very frequently across all instances in the dataset. Fig. 7.2 displays the features in each source against the number of instances in which that feature appears, with the belief that the most informative and discriminating features lay in-between the two extremes of appearing almost ubiquitously and appearing only once. From this hypothesis, KEGG pathways appear to have the most evenly distributed feature frequency, followed by GENNAV. Notably, these sources are query-integrated, in the sense that the features contained within them are never directly connected to the query itself, and thus are only reachable through proxy results *i.e.*, other sources, in the case of GENNAV, and KEGG genes for KEGG pathways.

7.2.2 Source-against-source performance

Using the methods outlined in Ch. 7.1.4, and with propagation weighting on the query graphs, SVMs were trained for individual queryable datasources. The optimum (by AUC) training parameters, per source and per label, were determined using a coarse, grid-based search on the C and σ for a Gaussian kernel. Execution of the resulting classifier on the test set are shown in Fig. 7.3.

Across all specific virulence categories the AUCs of the GENNAV and AMIGO datasources, whose records are indirectly queried from the seeding protein, outperformed all other methods and datasources, in some cases by very large margins. Examination of the sources as a whole additionally reveal that the data sources, individually, perform at varying levels. However, under Kendall's rank correlation, coverage appears significantly related to AUC at $\alpha = 0.01$ for three classes (Toxin, Surface factor, Defense), and $\alpha = 0.05$ at all others. Comparisons of the GO-based results to the other sources are further indicative that a learner based on integrated queries provides a more optimal classifier, and visibly by the greater convexity of the GENNAV and AMIGO curves. Undoubtedly, a strong contributing factor in this is the superior coverage afforded by these two sources; two other sources that are more lightly integrated, KEGG and INTERPRO also perform well relative to the other sources.

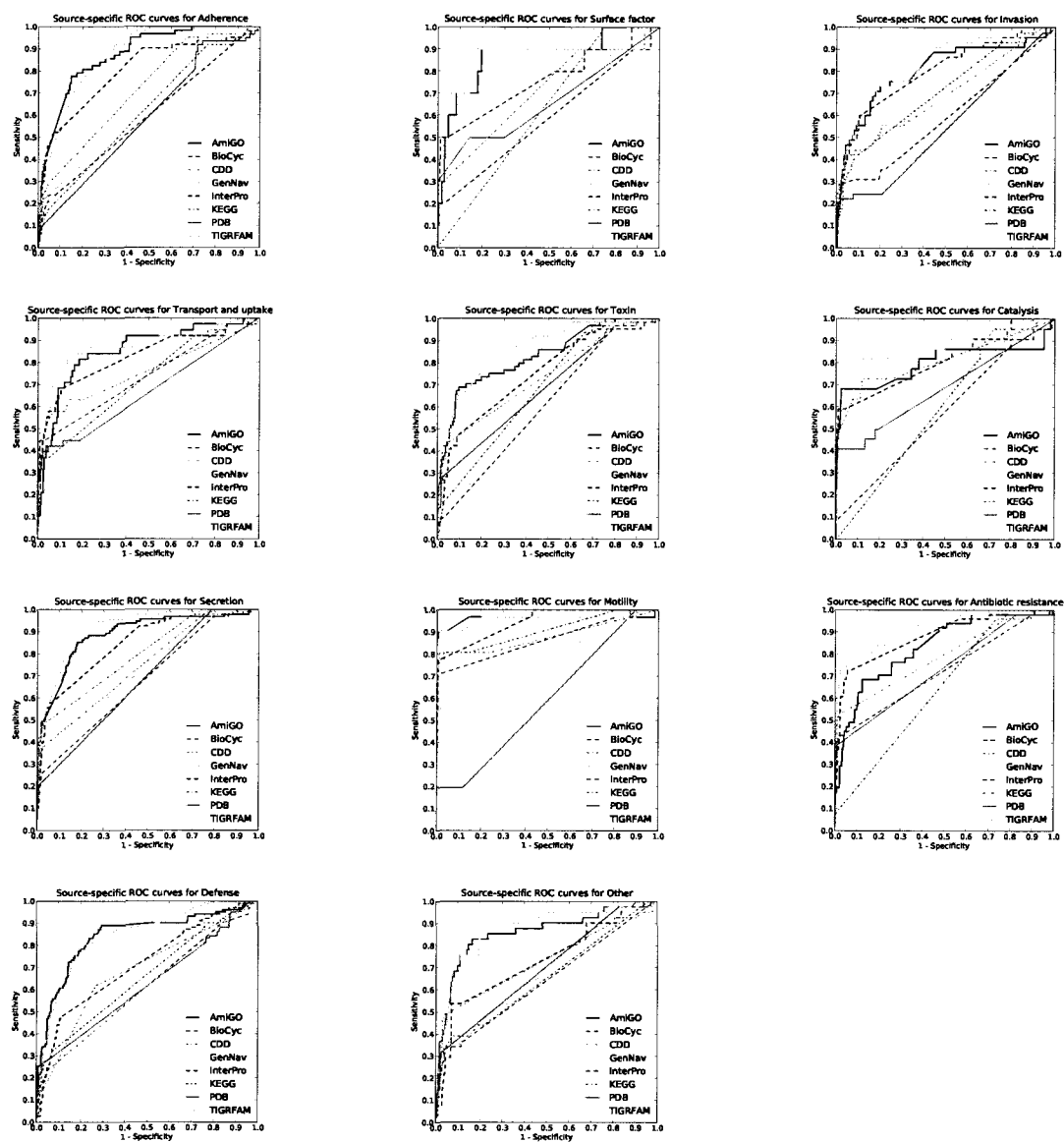


Figure 7.3: ROC curves for data-integrated sources using optimized parameters.

Further analysis of the ROC score results reveals other interesting results. Category 8, Motility, was relatively trivial to classify not merely by GENNAV but by other sources as well, including KEGG and INTERPRO. One explanation for these results is that the motility of pathogenic bacteria, and indeed bacteria in general, is a very well characterized process, and proteins related to bacterial motion are well-annotated and unambiguous. Despite its coverage in comparison to sources like TIGRFAM and BIOCYC, and contrary to the case in other categories, PDB fails to predict motility well. This may partly be due to the fact that motility-related proteins, given their high probability of containing transmembrane regions, are difficult to structurally elucidate and thus good exemplars of this class are more absent in this database.

7.2.3 Data integrated learning vs. baseline methods

Besides making inter-source comparisons, it was also important to compare data integration-based learning methods to baseline methods that have been previously published. As mentioned earlier in this chapter, several baseline methods were used in comparisons to query-level integration for learning specific virulence (see Ch. 7.1.4): a purely sequence-based approach (*3mer*) that uses an SVM classifier, a simple *k*NN method to emulate BLAST-based annotation and a pairwise BLAST-based SVM technique that has performed well in the past. Fig. 7.4 shows the pairwise comparison results of six five-fold cross validation runs with the sources and baseline methods, with better methods appearing higher in the graph. Note that unlike the parameter-optimized results in the previous section, feature selection based on training data for the sources was not performed and classification was done using an unpruned feature set. The metrics used in this experiment include the AUC, which has been used in the previous experiment, and the area under the ROC₅₀ curve, scaled to between 0 and 1 [185]. This truncated version of the ROC curve is the probability that a positive class will score higher than a negative class up to the first 50 false positives, and is often used in biology as a representation of the number of false positives a biologist might realistically be willing to manually review.

Statistical significance of pairwise comparisons are also visible in Fig. 7.4 *via* transitive

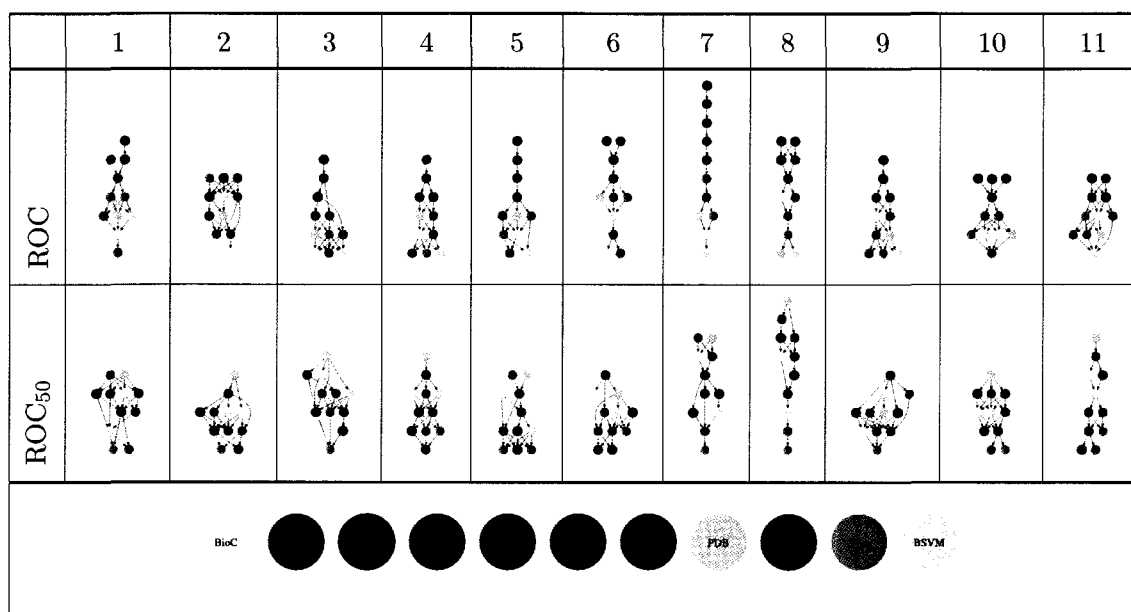


Figure 7.4: Statistical significances based on six five-fold cross-validations for all 11 virulence classes. An arrow from a head source or method to a tail source or method transitively indicates better pairwise performance from the head against the tail. Results, *via* two-tailed t -test, for ROC and ROC₅₀ are shown and nodes are color-coded based on source type (blue indicates domain- or motif-based sources, red GO term-based sources, yellow-brown for pathways, gray for structural sources and green for baseline methods). Actual values are listed in Tbls. B.1- B.13.

arrows. It stands out that in all but one virulence class, at least one of the queried data sources outperforms all baseline methods; *3mer* performance on the Surface factor label is exemplary compared to most sources and the other baseline methods. However, it is also the label with the fewest instances. In eight of the virulence classes, the ROC curves of GO-term based methods outperform not just baseline methods, but all other sources as well. Interestingly, for proteins related to antibiotic resistance, such as drug efflux pumps, INTERPRO does significantly better than all other sources and methods. Examining ROC₅₀ results provides a truncated view up to a specific number of false positives, and in this view

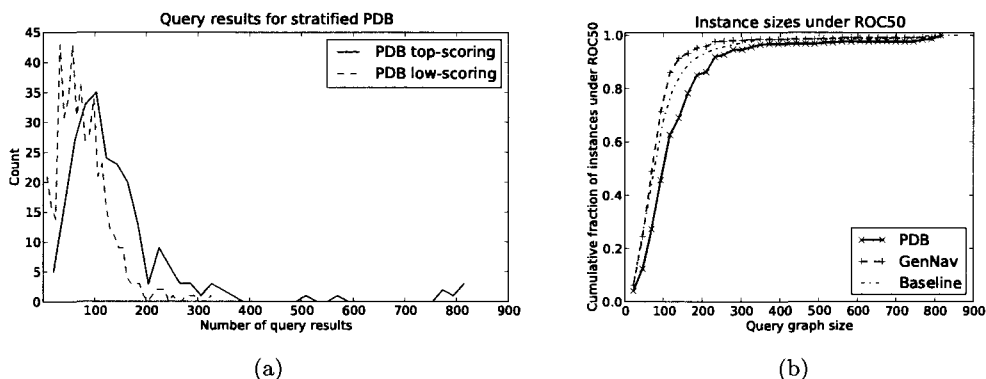


Figure 7.5: Fig. 7.5a shows histograms of query graph sizes for a subset of instances; those within PDB’s ROC_{50} curve (‘PDB top-scoring’) were generally larger than those outside of it (‘PDB low-scoring’). The difference between the two distributions (which appear shifted, though symmetrical) is statistically significant with a p -value less than $2.2\text{e-}16$, *via* two-tailed signed rank test. Fig. 7.5b shows cumulative probability distributions for the number of instances within the ROC_{50} of PDB and GENNAV against the baseline of all query graphs. PDB instances that fall under the ROC_{50} tend to be larger than both the overall average and GENNAV. These charts omit query graphs for which no results were returned.

the findings change. Sources closer to the queries themselves do better, and in particular PDB and TIGRFAM, whose performance under the full ROC was only at or similar to baseline methods. Moreover, the BLAST+SVM method, which is nearly indistinguishable from other baseline methods under the full ROC, does well compared both other baseline methods and sources that are more query-integrated, *i.e.*, AMIGO, GENNAV, INTERPRO and KEGG.

This was an interesting finding, and suggested that for some proteins integration may have added noise – a side-effect previously conjectured on from the experiments run for Ch. 5. To explore this prospect deeper, proteins in the optimized test set were stratified by those that fell under PDB’s ROC_{50} curve and those that were outside of it. The number of results for the two sets were reviewed, with the hypothesis that those that fell within PDB’s ROC_{50} curve would be better characterized, as measured by the size of its query

graph. The findings are reported in Figs. 7.5a and Fig. 7.5b, and suggest that for proteins that are already very well characterized, integration of further sources may be of marginal benefit. Notably, only 50% of query graphs that fall within the ROC_{50} of PDB are less than 100 nodes in size, whereas in the case of GENNAV 80% of the query graphs are less than 100 nodes in size. At the same time, proteins which are less well-characterized and more novel appear harder for an individual source such as PDB to classify; in such cases, integration across multiple sources as a means of providing coverage can alleviate the lack of information.

Because the proteins of interest for this dissertation are those involved in pathogenesis, it was also important to determine how well virulence classes could be discerned using varying levels of training set sizes. The motivation behind measuring this was to gauge how well-known a family of virulence factors may need to be for successful identification ‘in the wild.’ Fig. 7.6 displays the AUCs of a subset of sources (those that performed best in the generalized virulence classification test) and all baseline methods from three paired five-fold cross-validations under different and increasing training set sizes – 10%, 40%, 70% and 100% of the original training set sizes; testing sets remain untouched.

It immediately stands out that some classes are largely invariant to the number of instances seen, relative to the AUC. In the case of the Motility and Secretion classes, both KEGG and GENNAV perform essentially the same and with little change, though for other sources such as INTERPRO the AUC increased with the training set size. This leads to the conclusion that some data sources may better characterize classes than other sources, and that in the case of GENNAV and KEGG for Secretion, there are likely a set of terms or pathways that commonly describe pathogen secretory mechanisms, and that these annotations are widespread across the set of secretion-related proteins. Also of note is the performance of GENNAV under small training set size conditions. Other sources and methods generally tend to perform poorly (< 0.7 AUC) with training sets less than 500 instances, whereas GENNAV does considerably better in 7 of the 10 cases often by more than 0.1. This suggests that heavily integrated sources, such as GENNAV, may have additional utility over other methods when the number of seen and known instances from which to train are very low, furthering the finding that integrated methods do better under more ambiguous conditions.

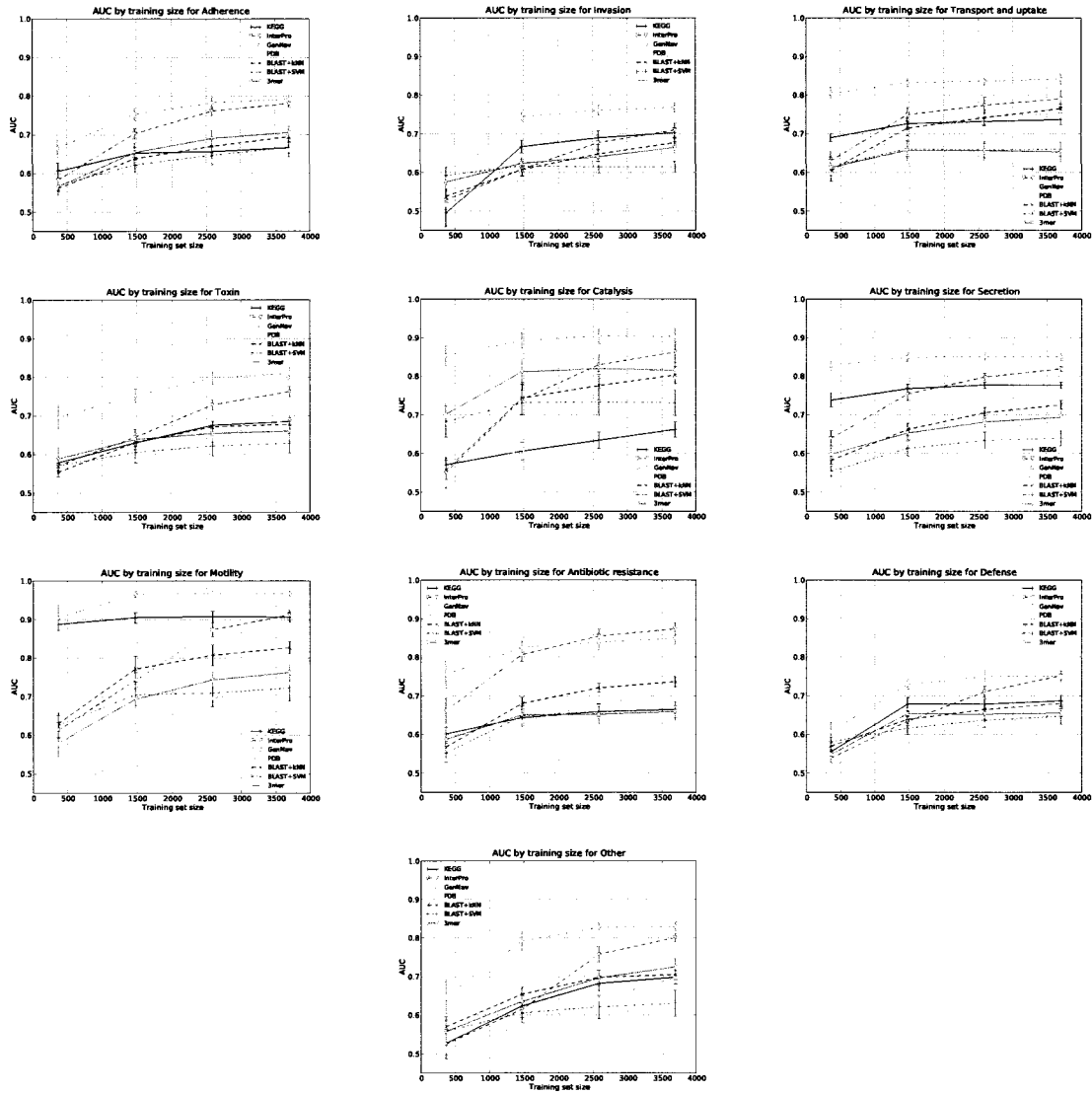


Figure 7.6: Average AUCs and 95% confidence intervals for a subset of sources and the baselines by training set size, based on three five-fold cross-validations. The ‘Surface factor’ virulence class is omitted due to the small number of instances present in the training set.

Thus far, results have strongly suggested that some categories are much easier to classify than others. For example, Motility is easily predicted across all integrated sources, and for some even at very low numbers of training instances. In comparison, other virulence classes such as Invasion and Defense remain harder to identify with strong confidence. Because of many of these nuances in performance are both source- and label-specific, it was of interest to generate ROCs using only the weights determined from the propagation algorithm in the query graphs. For each category and source, the F-score per (7.2) was computed, and the highest-performing feature was kept. Recall that each feature generated a weight from the propagation algorithm within the query graph (see 6.4); this value was used as the thresholding function for the generation of AUC scores. The result of this is a very basic classifier, $Top-F_1$, that relies only on the single-most discriminating feature of each source for each label. The AUC for this classifier represents upper bound (with respect to the harmonic mean of precision and recall) predictions ignorant of any value in combining multiple features; comparisons of this with other methods would thus illustrate any advantages or disadvantages from using more sophisticated approaches on the query graph data.

Fig. 7.7 shows a heatmap of the difference between SVM AUCs. Across all sources and labels, the ROC curves using SVM-based methods demonstrated added utility over $Top-F_1$, though colors trending toward the deep blue end of the spectrum represent only modest increases; colors closer to deep red are more marked improvements for the SVM method over $Top-F_1$. It is clear that there is marginal benefit to using a more sophisticated classification method for KEGG and GENNAV on the Motility class, although other sources such as BIOCYC derive a noticeable advantage from using SVMs for classification. This further corroborates the findings shown in Fig. 7.6, and for some sources and labels the mere presence of a single feature can be strongly indicative of membership.

At the same time, other sources benefit greatly from the combination of features, namely INTERPRO and CDD, the two sources which on average have the highest improvement of using SVMs over $Top-F_1$. At the opposite end, TIGRFAM and BIOCYC show the least improvement overall, with the GO-term based sources (AMIGO, GENNAV) showing moderate improvement. Examining the GO-term based sources shows interesting differences between

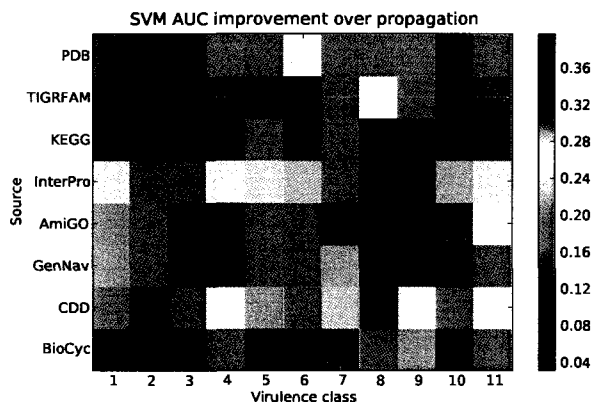


Figure 7.7: Improvement in AUC of using an SVM for classification for cross-validated tests over $Top-F_1$. The colors in each cell correspond to the difference in AUC between using an SVM for prediction and using the top-performing feature per source, per label. Higher values indicate cases where there is added benefit in considering multiple features *via* SVMs.

using only direct annotation information (AMIGO) and enrichment *via* traversing the GO hierarchy (GENNAV). While most of the changes between the two sources are commensurate, AMIGO strongly benefits from the use of SVMs for the Secretion category. One possible conclusion from this is that there are several top-level terms in GO that suggest secretion, and that under GENNAV these terms are retrieved; under AMIGO, however, this information is not available, but is ameliorated by the availability of terms that may share mutual parents.

7.3 Conclusion

Overall, the integrated methods perform significantly better than all baselines under the ROC metric for the greater majority of virulence categories. Under ROC_{50} performances are more muted, and though the most integrated sources outperformed the simpler baselines, sources closer to the query and the BLAST+SVM method did better across all virulence classes. One possible reason for this is that data scored in the ROC_{50} are those instances that score highest for the given method. These instances may represent ‘low hanging fruit’

that are already well-characterized, and extensive query-level integration merely adds noise. In cases where data is not available (*i.e.*, the proteins are less well-characterized), these low-coverage sources provide little information. The value in query-level integration may then be in providing information for harder-to-characterize, and more novel, sequences.

In consideration of the generalized virulence experiment, there was some slight change in source performance for specific virulence prediction. A notable finding was that the performance of AMIGO as a classifier was indistinguishable from that of GENNAV (see Fig. 7.3), which was not the case for generalized virulence. Since the difference between these sources is that GENNAV essentially recreates the subtree of the GO hierarchy within the query results, this may indicate that many of the GO terms specific to virulence categories share parents across other categories and complete traversal of the hierarchy has diminishing returns as it pertains to generating features for classification. At the same time, since the difference in generalized virulence between AMIGO and GENNAV was notable (see Tbl. 6.2) gives evidence to the belief that while different pathogenic processes may share similar functions overall, there is a more defined division between the functions occupied by non-pathogenic and pathogenic proteins.

One well-known drawback of using data integrated at the query level, using a federated approach, is the time-to-query; the time needed to gather the information necessary to run classification experiments on integrated GO term information was non-trivial, and ran at approximately four minutes per sequence on average, mapping to > 150 hours of query time for the specific virulence dataset. For other data sources that were directly linked to the query sequence (such as KEGG and CDD) retrieval would be satisfied at half that time, or less. Conversely, the baseline methods were relatively fast to compute, and in particular the k NN approach, whose total calculation was measured in minutes, excluding the time needed to build and query the associated BLAST database. Fortunately, this shortcoming scales well as the number of querying machines are increased and can be easily parallelized; at a modest four servers, for example, integrated GO term data could have been collected within 11 hours or less.

Also, the time spent querying is not fruitless, and independent of its application for classification the query results themselves may be of use to a biologist. Because one of the

features used in data integrated learning is a standardized vocabulary for describing gene function, individual classifier predictions can be easily reviewed and confirmed or rejected. Each query results in a number of GO terms that can be informative in determining why a particular protein was classified within a specific virulence category. Unless the biologist is very familiar with the mechanisms and characteristics of the proteins predicted to be related to virulence, it may be difficult to discern the biological relevance using other methods, such as the *3mer* approach whose features are only sequence word frequencies. Thus, in addition to providing classifications, the query-level integrated approach also acts to lessen the obfuscation often associated with learning methods.

Finally, despite very fruitful and interesting findings, there are several notable caveats with regard to both the dataset and the problem of specific virulence identification. Curation of the specific virulence dataset, and in particular the partitioning of proteins into the broad categories of virulence roles, was done in an iterative fashion and relied heavily on verification of experimental and literary evidence. Nonetheless, there is a strong possibility that a large number of proteins were misclassified or had correct classifications omitted; given the method used to curate evidence of virulence and the already difficult nature of virulence roles the latter is a more likely occurrence, though both are probable.

7.4 Discussion

In this chapter, the benefits of combining both data integration and classification were extended beyond generalized virulence to multiclass specific virulence. The added benefit of this versus the former problem is the ability to perform finer-grained predictions for virulence based on specific roles for the purposes of gene prioritization. Thus, there are two primary contributions made in this chapter to the dissertation as a whole, the first of which is the curation of a specific virulence dataset.

Thus far, virulence had been treated in computational classification and identification problems as a single category, despite the fact that there are often differences between various forms of pathogenic proteins which may occupy distinct, but perhaps overlapping, roles. This chapter presents a method and the resulting dataset of curated, non-redundant unified categorization of 11 different virulence factor classes and several other subcategories.

The formation and development of this dataset was based upon the disparate classification system of several virulence-related databases whose schemata, until now, had not been merged.

Second, this chapter provides the results of experiments in classifying proteins into the 11 virulence categories based on query results from data sources that were integrated in a path-based fashion. Additional experiments include the performance of one of these sources against several baseline approaches. Using the primary scoring metric, area under the ROC curve, it was found that GO term data sources significantly outperformed both other data sources and the baseline methods across a considerable majority of categories. However, under the more strict ROC_{50} metric, the results were more mixed, with well-curated sources more closely related to the query, such as PDB, performing best. The finding that no single source or method dominated in all classes across the all portions of the ROC curve suggests that in determining virulence using sequence-based homology queries, employing multiple data sources may provide a good hedge; when done using a data integration system, these queries can be made uniformly and in parallel, reducing scientist overhead in retrieving the data directly.

This chapter and the one preceding it, Ch. 6, focused on experimental results done within the confines of a controlled dataset. Though this provides convincing ‘laboratory-level’ results in regards to the utility of classifiers built on integrated data, it is also important to test the ability of the methods described on full, virulent proteomes. Coupled with manual inspection of the findings, the results of such predictions would provide not only practical validation of using query-level integration with learning methods, but also stand to contribute to biological research in the form of novel virulence factors within specific, pathogenic genomes.

Chapter 8

**GENERALIZABILITY OF QUERY- AND DATA-LEVEL
INTEGRATION AND LEARNING**

As previously seen in Ch. 7, the use of integrated queries is an effective means of forming a feature space upon which entities may be classified for detecting both general and specific virulence. However, the generalizability of this method within other problems in bioinformatics has not been discussed so far in this dissertation. The following chapter outlines another set of experiments to test the methods described in Ch. 3, 6 and 7 for use in a problem of more general interest for biomedical research.

8.1 *Applicability of methods and concepts*

One of the hypotheses that form the basis of this research is that integrated query graphs of interlinked data across fractured and heterogeneous sources add value, both as a method of expansive information retrieval and as a comprehensive feature space to which queries may be mapped. A common dichotomy in biology is that while experimental data is too few and far between, computationally-derived data is abundant and easily available. No single data source, however, provides absolute and perfect coverage for any given domain. Employing formal data integration methods thus acts as a means of extending coverage over queries of interest, and the use of statistical learning methods are an effective means of mollifying the distorting noise often seen in these databases.

8.2 *Case study: predicting generalized protein function*

A natural extension of the application for integrated queries is towards protein function prediction, a problem that is well-studied within literature (see Ch. 2.2 for a review), and a task which at first glance appears very similar to that of specific virulence role identification. However, two main aspects set this problem apart from specific virulence prediction. First, a protein's role in virulence is generally a product of its function, and is thus more of a latent

rather than directly-attributable label. Unlike virulence, protein function is often ascribed a definite status, and lines of demarcation between different functions are comparatively well-recognized. Second, whereas information in databases is not explicitly designed to capture virulence information (with some exception), protein function characteristics are much better captured in biological databases. Indeed, the GO hierarchy is intended to serve as a terminology of discourse around gene function.

While the differentiation may appear to favor protein function as an easier task, as alluded to in Ch. 2 it can be tremendously difficult. Ch. 5 showed, for instance, that functional data can be very noisy in spite of its abundance. Additionally, many automated methods of varying sophistication have shown to be quite effective first-pass annotation systems; improving on existing functional prediction methods is challenging. The following section outlines the methods and results of applying the integrated query as a discriminative feature space for learning specific protein function.

8.2.1 Methods

The sole datasource used for functional annotation from the integrated query graphs was GENNAV. This decision was grounded in the findings from both general (Ch. 6) and specific virulence prediction (Ch. 7), which show GENNAV to be a significantly better predictor than all other sources and baseline methods. Naturally, other sources were queried, however, as GO terms (and thus GENNAV) are only accessible from a protein query indirectly under the schema used (see Fig. 6.1).

For testing general function prediction, the data used for both training and testing was the functional protein set with the first 13 categories of MIPS for *S. cerevisiae*. This set of proteins is well-curated, and has been used extensively in the past for testing functional annotation methods [115, 17, 18, 186]. As was the case with the specific virulence dataset, proteins in the MIPS functional set could assume multiple labels, in addition to being a multiclass problem. Tbl. 8.1 lists the 13 MIPS categories, and the number of proteins that fall under each.

Like in the previous experiments, data querying and retrieval was done using the MIQAPS

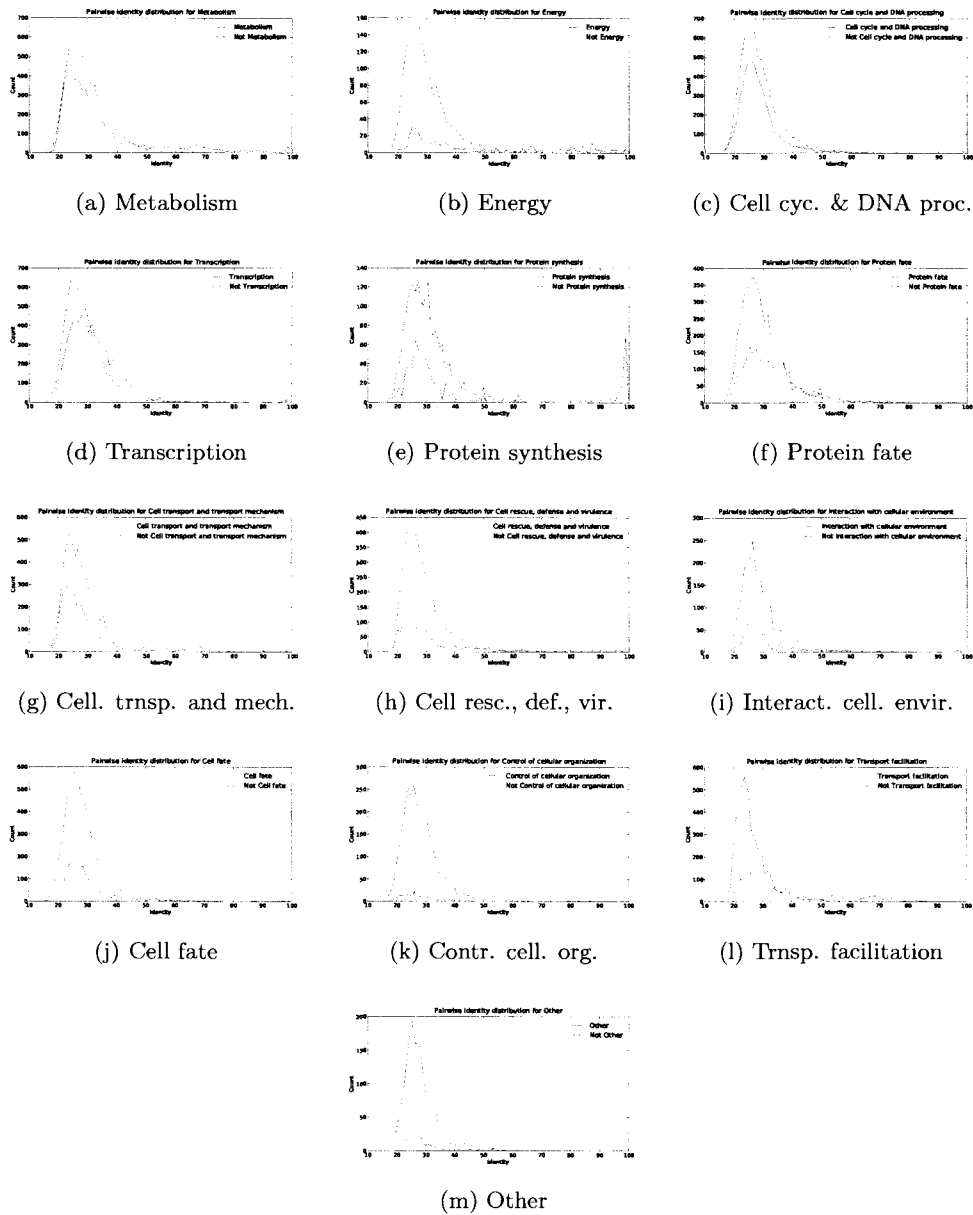


Figure 8.1: Inter- and intra-category similarity distribution by identity.

No.	Functional class	Count
1	Metabolism	1048
2	Energy	242
3	Cell cycle and DNA processing	600
4	Transcription	753
5	Protein synthesis	335
6	Protein fate	578
7	Cellular transport and transport mechanism	479
8	Cell rescue, defense and virulence	264
9	Interaction with the cellular environment	193
10	Cell fate	411
11	Control of cellular organization	192
12	Transport facilitation	306
13	Other	81

Table 8.1: MIPS functional categories, with protein counts drawn from yeast.

package and the propagation weighting approach was used to generate the initial values for nodes before kernel transformation. Classifier training and testing proceeded similarly to the baseline comparisons done in Ch. 7 *via* cross-validation. In these experiments, baseline performances were set by the results reported in the experiments by Lanckriet *et al.*, Deng *et al.* and Chua *et al.* [17, 115, 18], and the procedure for training and testing thus emulates the one outlined in [17] in both SVM parameters ($C = 1$ and $\sigma = 0.5$ for all classes) and test method in order to enhance comparability. Furthermore, the k NN method as described in Ch. 7 was added as a baseline, chosen for its simplicity; for this experiment, $k = 3$ was used. In this evaluation approach, five-fold cross-validation was done three times resulting in 15 separate ROC curves that were used to obtain the averages and standard deviations of the classifiers' performances.

8.2.2 Results

Relative coverage across all datasources was surprisingly slightly lower than the levels seen previously in the other experiments overall (refer to Tbl. 8.2). Given that the set of proteins originated from yeast, a model organism, one might have expected a larger number

of annotations propagated across genomes. Especially notable was the amount of data retrieved from BIOCYC and TIGRFAM, which were strikingly low in comparison to the other sources. Nonetheless, ratios of the number of unique records returned for GENNAV also approximated expected numbers, with 1.03 GO terms per protein on average.

<i>Data src.</i>	<i>Fraction of coverage</i>	
	<i>Fraction</i> ($n = 6355$)	<i>Source size</i> (in appr. units)
AMIGO	0.93	6580 terms
BIOCYC	0.05	351 proteins, pathways
CDD	0.56	6700 models
GENNAV	0.93	7733 terms
INTERPRO	0.73	4012 models
KEGG	0.92	213 pathways
PDB	0.40	8394 molecules
TIGRFAM	0.16	975 models

Table 8.2: Database coverage by fraction across all sources for the training and test sets by MIQAPS for the MIPS protein set.

The results of using integrated GENNAV in combination with an SVM classifier against the various baseline classifications are shown in Fig. 8.2; note that the results reported for the baseline method using semidefinite programming to optimize integrated kernel weights ('SDP + SVM') uses data which includes an enriched Pfam kernel with the Smith-Waterman algorithm (the highest-performing variant reported in [17]). Using integrated query graphs with a SVM classifier performs best across most of the 13 categories, though the difference is not statistically significant for some of the categories. Somewhat expectedly, the simple BLAST-based 3-NN baseline approach did well for the final category, 'Other', given the few assumptions that method makes and the varied nature of that category. Unexpectedly, however, this baseline approach did not perform much differently than the more sophisticated MRF method used by Deng *et al.*

Though it did not perform significantly different from GENNAV alone in the specific virulence case, a fused kernel of GENNAV and 3mer data was also tested, with weights

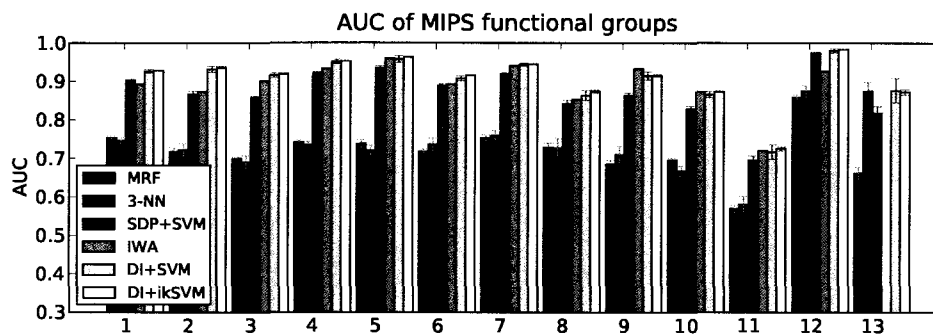


Figure 8.2: Comparison of ROC performance on 13 MIPS functional groups using Markov random fields (MRF), fused kernels with SVMs enriched with Pfam data (SDP+SVM), integrated weight-averaging (IWA), BLAST-based k NN, integrated query graphs with SVMs (DI+SVM) and integrated query graphs and $3mer$ in a fused kernel (DI+ikSVM). Error bars for denote 95% confidence intervals for average ROC, based on 15 cross-validated experiments. The data for MRF and SDP+SVM are from [17], and the data for IWA are from [18].

distributed equally (0.5) for both kernel matrices (see (3.7)). While the same conclusion on statistical significance to GENNAV could be reached in the instance of generalized function prediction, it was found that the fused kernel resulted in less deviation for ROC scores. Indeed, the fused query kernel (denoted in Fig. 8.2 as ‘DI+ikSVM’) showed performance above the non-fused query-level data integration approach for 12 of the 13 MIPS functional classes at a significant level. The sole exception, again, being the final catch-all category where a classification model with fewer assumptions did as well.

8.2.3 Discussion

The findings of this chapter generalize the methods and results of Ch. 6 and 7 by applying query-level data integration and learning to function prediction, as well as virulence. Notably, the sources and schema used for this generalization are the same as those used for the virulence experiments, and suggest that this basic data model oriented around a selected number of sources to provide domain, pathway and structural coverage may be of broad

utility for tasks in molecular biology where protein sequence queries are involved.

For this task, the usage of a naïve integrated kernel composed of query graph data and *3mer* data had a stabilizing effect on the variability of the results; these are interesting findings, and suggest that the addition of other sources or a more finely-tuned weighting may eventually provide better performance, as was demonstrated in previous work [17]. A shortcoming of this approach, however, is the computational demands for assembling and learning on integrated kernels – attempts to integrate three large query kernels led to excessive paging on a machine with 2GB of memory. Strategies for feature selection, or alternatives to kernel-level integration, may be needed to make this approach more scalable across a large number of datasources.

Nonetheless, even using a single query-integrated source (GENNAV) produced good results for protein function prediction, particularly considering that this method is entirely sequence similarity-based, versus the compared baseline methods which used protein-protein interaction and experimental data. However, one caveat to these findings is that as the baseline methods were conducted in the past, it is guaranteed that information within the datasources used for this experiment have increased and improved since; it is very likely that performance would be different had this experiment been conducted earlier. At the same time, this also serves as a strong argument for methods that can scale with the pace of information growth. In this regard, query-level data integration is an attractive option.

Chapter 9

CONTRIBUTIONS AND SIGNIFICANCE OF WORK

The preceding chapters have posed a problem of biological importance and have proposed and tested a combination of computational methods as a means of addressing it. In Ch. 1 and 2, the challenges of identifying and elucidating pathogenic proteins is outlined, while Ch. 3 and 4 describe data integration and statistical learning as a set of approaches that can serve to address the challenges. They further introduce a system capable of uniformly querying biological data sources in a path-based, federated fashion under different models of data retrieval suitable for biology. Ch. 5 provides a statistical characterization of the nature of integrated and heterogeneous biological data as it pertains to the problem, and motivates the reasoning behind using high-coverage methods as a means of producing predictive features for classification algorithms. This is followed by the experiments of Ch. 6 and 7 that empirically demonstrate query-level integration with supervised learning is capable of predicting pathogenic proteins, and at levels better than other approaches used as baselines. Ch. 8 then illustrates the generalizability of the methods described in this dissertation beyond the specific problem of pathogenicity to generalized function prediction, where it performs well against other methods, including those that are state-of-the-art.

This final chapter reviews these findings in summary, and specifically highlights the main contributions made by this dissertation. Biologically-relevant implications of the findings are outlined, and limitations of the methods and results are also discussed within this context. Finally, specific areas which are left unresolved in this dissertation or are of tangential interest are raised as future work.

9.1 Novelty of path-based federated integration and learning

Early on, two major rate-limiting steps in biological research are the lack of available information regarding a protein, and the lack of time and resources for manual curation of data.

The formal framework for data collection and integration posited by this research moves towards a direction where biological information can be uniformly and easily queried without having to explicitly handle idiosyncrasies of individual repositories. As demonstrated by the experiments described in this dissertation (see Ch. 6 and Ch. 7), the task of deriving knowledge from scientific information can be alleviated using learning methods that discriminate between proteins involved in different virulence-related roles. If such classification schemes are robust to the inevitable amount of noise encountered in biomedical datasets, the automated annotations resulting from this retrieve-classify process can effectively aid scientists in guiding their research and highlighting targets of possibly high impact for experimental assay or manual curation.

The methods of this dissertation attempt to reach this level of utility, and importantly tries to do so using publicly available data that is known to be both cheap and abundant. Whereas other previous research has attempted to use such biological data as an end to itself by careful and principled means of automated and semi-automated curation [20, 36, 37, 101], one notable aspect of this research is the side-effect of bypassing this goal entirely. This dissertation's results, produced by combining path-based federated integration and learning, are not necessarily dependent on the quality of the data contained within the individual repositories; indeed, one assumption made regarding the data is that for any given protein query, irrelevant data will be present, but that such information may still be useful in classification if such irrelevancies are also present in proteins of the same class.

The generalizability of this research was tested on traditional function prediction, a well-studied area of biological and computational research (see Ch. 8). Using a dataset relied upon in the past many times for function prediction, results from this experiment were encouraging as the performance of query- and data-level integration outperformed other methods in most protein functional classes, and in many cases by a significant amount. Notably, the schema and sources used for this generalization were the same as the virulence cases, suggesting that schemata across biological tasks may have some limited form of shared utility. As in the case of the virulence datasets, the feature inputs for classification were drawn from myriad data sources.

In addition to the aforementioned characteristics of functional coverage and class discern-

ment, an additional benefit of the combined federated and learning approach presented by this dissertation is in the interpretation of findings. For many biologists statistical methods, such as SVMs, remain a ‘black box’ technology – the intimate operations of the technique can be difficult to understand, and the process by which results are arrived are often unclear. By basing predictions on functional biologic information such as GO terms, scientists may examine the findings of the classifications and then retrospectively review the data gathered that may have contributed to the results. For example, among the most indicative GO terms for the ‘Antibiotic resistance’ specific virulence class (by F-score) were `GO:0046677, response to antibiotic` and `GO:0008800, beta-lactamase activity`, the former being a self-evident finding and the latter a well-known biological process associated with antibiotic resistance. This is in contrast to prior work, where functional information may be harder to elucidate from the findings, and thus results may be less penetrable for the average biological researcher [95, 115].

Adequate coverage of such functionally informative terms would be much more challenging without applying integration of sources at the query level. Moreover, triangulation of multiple sources is not an exotic concept to biomedical researchers, as many automated annotation systems perform this to some degree already in an *ad hoc* fashion (see Ch. 2.1.2); the system and method described in this dissertation formalizes this and attempts to translate it into a computable form for learning that retains information regarding cardinality, topology and lineage. Along this line, the use of a path-based federated data integration architecture for retrieval was a key contribution employed in this research (see Ch. 4).

Considering the application of learning to path-based federated data in itself, one can imagine three factors of primary importance for deriving biological information: cost, quality and scalability. The chart shown in Fig. 9.1 displays the methods presented in this research (shown as *A*) in the context of these three considerations, and relative to two other broad approaches that have been used in the past in conjunction with path-based data integration. A decided advantage of a supervised learning approach, as employed for combined data integration and learning, is that the task of classification scales well as new data is introduced; the production of a new, adapted classifier is a relatively straightforward process, with most cost being attributable to parameter and feature selection. It must be

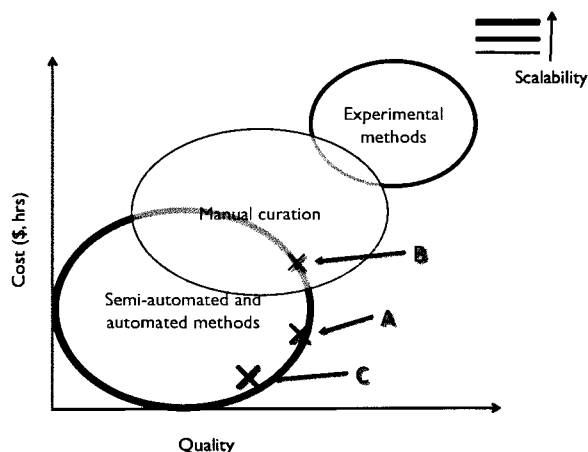


Figure 9.1: Perceived comparison of three path-based federated approaches (*A*-discriminative, the method of this dissertation; *B*-logical, as described in [19]; *C*-belief, as described in [20, 21]) to deriving biological knowledge along three axes (cost, quality and scalability). These methods are juxtaposed against the backdrop of the cost, quality and scalability levels traditionally associated with semi-automated and automated methods, manual curation methods and methods that produce experimentally validated evidence.

noted, however, that there is some considerable upfront cost in labeling a dataset with the classes of interest, if such dataset is not already available; this was a task undertaken for this research for specific virulence. For a logical model (shown as *B*), existing rules may need to be changed or new rules added as data evolves. Though they can closely emulate the quality of live researchers, because these rules are optimally derived from expert knowledge, their development and curation is an expensive and time-consuming process that does not scale as well when new information is discovered. There are thus considerable associated upfront and maintenance costs, and this approach does not scale well to growing biological data. On the other hand, belief-based methods (shown as *C*) of the type used in [20, 21] are perhaps the cheapest to employ, scale well as values are determined directly from the contents of the query graph and are fairly robust even with naïvely set parameters. However, as was demonstrated in [187], this avenue suffers from considerably decreased quality

in comparison to the more expensive logical approach.

Further work would be required in order to validate generally how well these different theoretical approaches fit within the above paradigm of cost, quality and scalability. Within the scope of automated molecular biology research, however, it is clear that methods of data integration at the query level are a rich source of informative, albeit disparate, material with which to base annotations and classifications. As was shown in this work, coupling this data with statistical learning provides a scalable and precise method for target prioritization.

9.2 Novel biological implications

The work of this dissertation is grounded on the overarching theory that a multitude of biologic data drawn disparately and of varying relevancy can be used to generate high-quality classifications, and there are many subsequent practical applications. As alluded to in Ch. 2, foremost among these applications is proteome-wide prediction of virulence factors, and facilitating selection of proteins for further study that have a high likelihood of being involved in infectious pathogenesis.

As an illustration of this, the methods of this dissertation were applied to the prediction of virulence proteins in *Burkholderia pseudomallei*, the causative agent of *melioidosis*. The disease is often fatal, particularly in the septic form, and even in treated cases has a 40% mortality rate [188]. Tables 9.1a and 9.1b provide an overview of the results obtained from submitting the entire proteome of *B. pseudomallei* into the specific virulence prediction classifiers generated from Ch. 7. To ensure high-specificity classification, only those proteins whose decision threshold was at or above the 95% precision level for each virulence category (based on cross-validated results over the entire specific virulence dataset) were selected; parameters were non-optimized. This represented a conservative, high-confidence, set of proteins which are possibly involved in virulence, and though comprehensive analyses of proteome-wide results are left for future work, many of the top-scoring classifications appear plausible, given the actual protein annotations.

The examples provided give some hint at the possible biological implications of the information generated from using data integration and learning for virulence recognition. Researchers interested in a specific organism or disease may use the class outputs and

<i>Class name</i>	<i>#</i>
Adherence	22
Surface factor	17
Invasion-related	18
Transport & uptake	77
Toxin	36
Catalysis	8
Secretion	90
Motility	43
Antibiotic resistance	147
Resistance & defense	34
Other	0
<i>Total labels assigned</i>	492
<i>Total unique proteins</i>	432

(a)

<i>TREMBL ID</i>	<i>Annotation</i>	<i>Predicted role</i>
Q3JS28	Type-1 fimbrial protein, A subunit	Adherence
Q3JI69	Phospholipase C	Toxin
Q3JHL6	Hemin transport protein HmuS	Transport & uptake
Q3JTT3	Dihydropteroate synthase	Antibiotic resistance
Q3JXA5	Response regulator protein	Invasion
Q3JLM7	Prolyl oligopeptidase family protein	Catalysis
Q3JP63	Cap. polysac. biosynth. protein fam.	Defense
Q3JN46	Unnamed protein product	Defense

(b)

Table 9.1: Tbl. 9.1a shows the number of predicted virulence proteins in *B. pseudomallei*, via data integration and SVM learning. GENNAV GO terms were used as the predictive features, and the threshold was set by 95% precision of the specific virulence dataset. Tbl. 9.1b is a sampling of the top-ranked possible virulence factors from the predictions.

scores of the system as guide to which proteins are most likely relevant to their research. For example, ‘Hemin transport protein HmuS’ (shown in Tbl. 9.1b) is a protein involved in heme transport. While this protein (Q3JHL6) has not been directly implicated in the virulence of *B. pseudomallei*, recent studies have shown that in low-iron conditions, not unlike that encountered in a human host, heme-related transporters and receptors are up-regulated in *B. pseudomallei* [189]. In this organism, HmuS may be a tempting target for possible attenuation of infection severity.

Examples as the one just discussed illustrate how the results of the methods described in this research can help guide the attention of scientists to targets of interest. Overall, as displayed by the findings for both generalized and specific virulence, this work provide a means of detecting, classifying and ranking proteins that may be of important health-related impact. Moreover, the data which drives the procedure do not come from expensive

experimental methods and is publicly available to any research biologist. Ch. 4 describes the process by which this data can be uniformly and easily gathered without regard to the individual repositories themselves. Together, this provides a complete system usable by biologists vertically integrated from gathering initial data on a protein to the end result of prioritization (and beginning of experimental or manual review) for virulence-relatedness.

9.3 Limitations of methods and results

Despite the performance of combined data integration and learning for various tasks, and the overall generalizability of the approach to other domains of biology, there are several limitations of which a reader should be aware when regarding the findings of this research. While these methods work well for both virulence and traditional function prediction, there is little guarantee that performance would be similar had different schemata been tried. The schema used was developed specifically to reach functional information in the form of GO terms. It is possible that another schema not so specialized and which converged to different data sources may have produced vastly different results.

Furthermore, it is difficult to predict how well the classifiers generated as a result of training and testing for the different classification experiments will generalize as the data sources in the schema evolve over time. Because the training process and the curation of newly labeled ‘gold standard’ data is the most time-intensive of combining query-level integration and learning, the ideal situation would be that the quality of the classifiers generated are largely invariant to most changes encountered in databases. And, for simplicity, this work has assumed that the data sources and the information within them remain relatively unchanged.

Indeed, some evidence from Ch. 5 indicate that for functional annotation the same terms will be encountered over and over again; and results from Ch. 6.2.3 show that the individual changes in databases separately do not seem to make a difference in using integrated data for prediction, as topological information on the query graph appears of only marginal benefit within this context. However, practically speaking, as information is added, modified and deleted in the repositories the information that is retrieved will change. These changes may slowly shift the content of databases across a domain; database updates in this manner may

well affect results substantially, and methods of maintaining the currency of classifiers in the face of constantly changing data is an interesting and unexplored aspect of this research.

One advantage of growing databases, though, is that further information would also increase the coverage afforded by a formal integration for data retrieval. For the experimental results of this research, GO functional coverage was reached for almost 90% of all proteins – an impressive number, given that the level of functional information available for any given genome can range from 90% to 50% or less (see Tbl. 2.3). A strong limitation of data integration and learning, however, remains the amount of information that can be obtained for any given query. For this work, sequence similarity searches were used to gather data for a query sequence. Under this model, an entirely novel protein for which no close homologues exist would be unclassifiable. While there are many ways of circumventing this problem, such as relaxing similarity requirements so that even very dissimilar proteins are returned, it is unclear how these compromises for coverage, at the expense of quality, would affect the results. The BLAST+SVM baseline method described in Ch. 7.1.3, which produces a pairwise kernel based on query similarity to arbitrary proteins given a very high expect value threshold, is this circumvention for a single source and performed relatively well in comparison to other baseline methods. However, given the better ROC performance of many other data sources (*e.g.*, CDD, INTERPRO, AMIGO) it seems that there is value in keeping the information used for prediction as relevant to the query as possible.

From a perspective of validity, a further limitation is the way in which the specific virulence dataset was curated. Though every effort was made to ensure the classifications used were of high quality and specificity, the proteins and classes used in Ch. 7 were conducted only by a single person, the author of this monograph. Curation was done in an iterative fashion (see Fig. 7.1) and thus proteins were essentially ‘classified’ multiple times until convergence. Undoubtedly, there are proteins either whose correct label has been mistakenly omitted, an incorrect label assigned or both. Based on the procedure used for classification, an error of omission is more likely than that of commission – a challenging problem for the development of any gold standard dataset in a field where scientific knowledge is in a state of constant expansion. As a result, for the specific virulence dataset a perfect classification may in fact be impossible, given differences the level of

comprehensiveness of classification for each protein. At the same time, this important limitation acts mostly as an upper bound on experimental performance and should not call into question the relative standing of the approaches compared; any errors in classification are independent of each other and uniformly distributed amongst the training, test and cross-validation sets.

Additionally, the virulence datasets consisted entirely of bacterial microorganisms, and thus the ability of the approach to discern proteins of pathogenic eukarya were not represented or tested. Attempts to apply bacterial virulence learning models to eukarya have met only modest success [95], and it is likely that the methods described in this research will produce less than optimal results when applied to eukaryotic or viral pathogens. Nonetheless, extending these methods to other kingdoms would be relatively easy; likewise, narrowing the granularity, by adapting the methods to specific phyla or families for example, would be a matter of curating virulence factors restricted to that set. In the former case, a looming challenge may be the number of known virulence factors available for a specific subset of organisms, and in both cases it would be advisable to generate new datasets specific to each.

Perhaps the most important limitation is that this research has focused primarily on only one aspect of pathogenesis – that is, the proteins in the disease-causing organism itself. For this work, the goal at hand was target prioritization for proteins within pathogens most likely to be involved in virulence. As previously mentioned in Ch. 2.1.1, the characteristics of the host are also of great importance, and recent examinations of the interactions between host and pathogen at a genomic level have highlighted the fact that a subset of well-connected human genes are the targets for many viruses [190]. Subsequently, an interesting corollary to the methods of this dissertation may be to incorporate host information to determine the manner in which predicted virulence proteins interact with human protein targets, improving knowledge of how deletion or mutation of specific infection-related proteins may result in attenuated virulence.

9.4 Future work

Despite the role their usage has played in making many remarkable advances in biomedicine, the use of publicly available repositories for high-throughput analyses is a largely unexplored

field. The preceding chapters of this dissertation have outlined, enumerated and answered the method and hypothesis of whether or not these sources can be mined collectively to generate biologically-relevant virulence knowledge. At the same time, the findings have revealed other interesting, tangential questions that remain to be answered, and many challenging problems that have yet to be solved.

One obvious question is whether or not there are better ways of combining heterogeneous and disparate data such that virulence predictions are improved. The virulence datasets, notably, were tested foremost using SVMs, and more broadly discriminative classifiers, whereby classes are learned directly from the data. Thus, the results from experiments are subject to the limitations well-known for maximum margin-based methods, such as performance sensitivity to parameters. The choice of a discriminative approach is in contrast to Bayesian, or generative methods, which attempt to model the likelihood that any given instance fits a class using probabilistic means [191]. Adoption of a discriminative angle was predicated largely by concerns on the spuriousness of information that may be present in the integrated data; SVMs in particular have been known to be accurate even for very noisy data, and whose applications to classification tasks in biological domains have often equaled or outperformed other methods (*e.g.*, [192]). Nonetheless, there are many compelling reasons to explore probabilistic approaches to virulence detection, not the least them being the manner in which the data is represented (graphically) – a structure which lends itself well to methods relying on dependencies. Indeed, based on the findings for virulence, it is likely that probabilistic methods of data fusion on the combined data sources could yield promising results.

Moreover, while integrated kernels of disparate sources was only briefly explored for traditional function prediction (see Ch. 8), other means of explicitly combining sources that are perhaps less computationally expensive may be interesting. A possible avenue of research in this area is to explore various stages of source integration (*e.g.*, early integration) but with restrictions imposed by logical queries of the type definable by query languages such as DaRQL. This would allow bench scientists to formulate what they believe to be the most trustable data within the myriad returned, with the hypothesis that the more relevant data will be less noisy and more predictive.

Future work should also include extending the results of this research beyond virulence and generalized function prediction to other domains, including those outside molecular biology. A interesting implication from this would be the generalization of query-level data integration and learning across varied schemata, and a possible way of validating this may be to test different schemata and changes in source inclusion across the same classification problem.

And an aspect of this research which was neglected, but is of paramount practical importance, is to explore how best to present the results of classifications based on multiple sources to a biologist. Any ground-breaking *in silico* findings for biomedical research are for naught if the information cannot be easily digested by those who do the experimental validation. Ease of use is a strong reason to include various querying paradigms (*e.g.*, exploratory, declarative) within any data integration framework, and similar parallels exist when trying to present actionable information to biologists from statistical methods. Comparatively little has been done in this area, where biomedical computation meets comprehensible presentation, and given the amount of uncertainty and data often dealt with by researchers it will continue to remain an open problem.

9.5 Concluding remarks

One goal of this dissertation was to rely on abundant, error-filled, messy data to generate predictive findings that rival those of other methods that might rely on more well-curated or even experimental inputs. Methods that scale with the growth of data are now more than ever important in biomedical research, and it may not be folly to presume in the very near future that the computer will be as commonplace a reference in biomedical research as the lab notebook, or algorithms as well-used as a pipette. Regardless of the methods, the most integral piece of solving biological puzzles will be the data, and part of that are means of effectively integrating, mining and navigating its nuances. Leveraging this deluge of data, despite many imperfections, can yield insight which might otherwise take considerably more time to expose itself.

By presenting a system and method of mining this data in the context of two biological tasks and presenting the results, a major overall contribution made by dissertation is that

even without extensive manual curation integrated data can be very effective for prioritization of both virulence factors and general function prediction. This approach scales well against both the number of sources incorporated and the amount of ground truth information known, making it an appropriate choice for high-throughput biological research. Lastly, any results produced by the artifacts of this research are possible targets of health-related interest in a new and evolving subfield of public health and biology; highlighting these proteins for additional study may further improve knowledge of pathogenesis and disease.

Appendix A

INTEGRATION ARCHITECTURE

A.1 Query graph definitions

This section contains the complete query graph data definition used for this dissertation. Comment lines are preceded by a ‘;’, and the block is divided into the following ordered sections: the schema itself, source declarations and link definitions. Each section begins with a (shorthand) illustration of how each data type is declared.

```

===== BEGIN BLOCK =====
; Copyright (C) 2008 Eithon Cadag
;
; This program is free software: you can redistribute it and/or modify
; it under the terms of the GNU General Public License as published by
; the Free Software Foundation, either version 3 of the License, or
; (at your option) any later version.
;
; This program is distributed in the hope that it will be useful,
; but WITHOUT ANY WARRANTY; without even the implied warranty of
; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
; GNU General Public License for more details.
;
; You should have received a copy of the GNU General Public License
; along with this program. If not, see <http://www.gnu.org/licenses/>.
;

; Bioinformatics-oriented MIQAPS schema to run with PyDI

; :cls -- top-level: class declaration; o.w.: class reference
; :isa -- class inheritance
; :atr -- class attribute
; :src -- top-level: source declaration; o.w.: source reference
; :trg -- class instantiation trigger
; :prp -- property declaration or reference
; :map -- mapping from local label to global attribute (within a class)
; :qry -- query declaration within a class

```

```
; :par -- parameter passable to whatever carries out the query
; :lnk -- link declaration
```

```
;;;;;;;;;;;;; Schema ;;;;;;;;;;;;;;
```

```
; (:cls <name>
; ( {(:isa <name>)} )
; ( {(:atr <name> .|*)} ))
```

```
(:cls Entity
```

```
()
```

```
())
```

```
(:cls Query
```

```
((:isa Entity))
```

```
((:atr QueryString .)))
```

```
(:cls ProteinSequenceQuery
```

```
((:isa Query))
```

```
()
```

```
(:cls GeneNameQuery
```

```
((:isa Query))
```

```
()
```

```
(:cls SourceQuery
```

```
((:isa Query))
```

```
((:atr Sequence .)))
```

```
(:cls Feature
```

```
((:isa Entity))
```

```
((:atr Desc .)
```

```
(:atr Start .)
```

```
(:atr Stop .)))
```

```
(:cls Protein
```

```
((:isa Entity))
```

```
((:atr Name .)
```

```
(:atr Species .)
```

```
(:atr Sequence .)))
```

```
(:cls Family
```

```
((:isa Entity))
```

```
((:atr Name .)
```

```
(:atr Desc .)))
```

```
(:cls SuperfamilyCls
```

```
((:isa Family))
```

```
()
```

```
(:cls Domain
```

```
((:isa Entity))
((:atr Name .)
 (:atr Desc .)))
(:cls Gene
 ((:isa Entity))
 ((:atr Species .)
 (:atr Desc .)))
(:cls PseudoGene
 ((:isa Gene))
 ((:atr Degrade .)))
(:cls Attribute
 ((:isa Entity))
 ())
(:cls CrystallizabilityAttribute
 ((:isa Attribute))
 ((:atr Crystallizability .)))
(:cls Structure
 ((:isa Entity))
 ((:atr Name .)))
(:cls Term
 ((:isa Entity))
 ((:atr Name .)
 (:atr Desc .)))
(:cls HierarchicalTerm
 ((:isa Term .))
 ((:atr Parents *)))
(:cls SignalPeptide
 ((:isa Feature))
 ())
(:cls Pathway
 ((:isa Entity))
 ((:atr Name .)))
(:cls TransmembraneRegion
 ((:isa Feature))
 ())
(:cls SequenceComposition
 ((:isa Attribute))
 ((:atr MetPct .) ; M
 (:atr LeuPct .) ; L
 (:atr AsnPct .) ; N
 (:atr IlePct .) ; I
 (:atr LysPct .) ; K
```

```

(:atr GluPct .) ; E
(:atr AspPct .) ; D
(:atr GlyPct .) ; G
(:atr AlaPct .) ; A
(:atr CysPct .) ; C
(:atr HisPct .) ; H
(:atr PhePct .) ; F
(:atr ThrPct .) ; T
(:atr TyrPct .) ; Y
(:atr ProPct .) ; P
(:atr TrpPct .) ; W
(:atr ArgPct .) ; R
(:atr SerPct .) ; S
(:atr GlnPct .) ; Q
(:atr ValPct .) ; V
(:atr PositiveChargePct .)
(:atr NegativeChargePct .)
(:atr TotalChargePct .)
(:atr NetChargePct .)
(:atr HydrophobicsPct .)))

```

```

;;;;;;;;;;;;; Source declarations ;;;;;;;;;;;;;;

```

```

; (:src <name>
; ( {(:cls <name>
; (:trg <label>
; ( {(:prp <name> .|*)} )
; ( {(:map <label> <name>)} )
; ( {(:qry <name>
; ( {(:prp <name> .|*)} )
; ( {(:par <name> <value>)} ) } ) } )
(:src Seed
(:cls ProteinSequenceQuery
(:trg pseq_query)
((:prp Seeder2NCBI .))
((:map src #SOURCE)
  (:map id #ID))
(:qry SeederQuery
()
()))
(:cls GeneNameQuery

```

```
(:trg gname_query)
((:prp Seeder2NCBIGene .))
(:(map gene_name Seeder2NCBIGene)
 (:map src #SOURCE)
 (:map id #ID))
((:qry GeneQuery
 ()
 ())))
```

```
(:src Superfamily
 (:cls SuperfamilyCls
 (:trg superfamily)
 ()
 ((:map id #ID)
 (:map src #SOURCE)
 (:map name Name))
 ((:qry SuperfamilySequenceQueryable
 (:prp Evaluate .))
 ())))
```

```
(:src TIGRFAM
 (:cls Family
 (:trg family)
 ((:prp Function .)
 (:prp TIGRFAM2EC *)
 (:prp TIGRFAM2InterPro *)
 (:prp TIGRFAM2GO *))
 ((:map id #ID)
 (:map src #SOURCE)
 (:map name Name)
 (:map function Function)
 (:map ec TIGRFAM2EC)
 (:map interpro TIGRFAM2InterPro)
 (:map go TIGRFAM2GO))
 ((:qry TIGRFAMQueryable
 (:prp Evaluate .)
 (:prp Score .))
 ())))
```

```
(:src BioCyc
 (:cls Protein
 (:trg protein)
 ((:prp BioCyc2GO *)
```

```

(:prp BioCyc2NCBI *)
(:prp BioCyc2UniProt *)
(:prp BioCyc2Pathway *)
((:map id #ID)
 (:map src #SOURCE)
 (:map name Name)
 (:map biocyc2go BioCyc2GO)
 (:map biocyc2ncbi BioCyc2NCBI)
 (:map biocyc2uniprot BioCyc2UniProt)
 (:map biocyc2pathway BioCyc2Pathway))
((:qry BioCycSequenceQueryable
 (:prp Evaluate .)
 (:prp Score .)
 (:prp Length .))
 ()))
(:cls Pathway
 (:trg pathway)
 ()
 (:map id #ID)
 (:map src #SOURCE)
 (:map name Name))
((:qry BioCycPathwayQueryable
 ()
 ())))

(:src KEGG
 (:cls Gene
 (:trg gene)
 (:prp Gene2Term *)
 (:prp Gene2Pathway *)
 (:prp KEGG2EntrezGene .)
 (:prp KEGG2UniProt .))
 (:map id #ID)
 (:map src #SOURCE)
 (:map org Species)
 (:map def Desc)
 (:map 2orthology Gene2Term)
 (:map 2pathway Gene2Pathway)
 (:map kegg2ncbi KEGG2EntrezGene)
 (:map kegg2uniprot KEGG2UniProt))
 (:qry KEGGSequenceQueryable
 (:prp Evaluate .)

```

```

(:prp Bits .))
()))
(:cls Term
(:trg term)
((:prp Term2Pathway *))
((:map id #ID)
(:map src #SOURCE)
(:map name Name)
(:map def Desc)
(:map 2pathway Term2Pathway))
((:qry KEGGOrthologyQueryable
())
()))
(:cls Pathway
(:trg pathway)
()
((:map id #ID)
(:map src #SOURCE)
(:map name Name))
((:qry KEGGPathwayQueryable
())
()))))

(:src SAPS
(:cls SequenceComposition
(:trg seqanal)
((:prp FIKMNYGroup .)
(:prp STGroup .)
(:prp AGPGroup .))
((:map src #SOURCE)
(:map fikmny_grp FIKMNYGroup)
(:map st_grp STGroup)
(:map agp_grp AGPGroup)
(:map pos_charge PositiveChargePct)
(:map neg_charge NegativeChargePct)
(:map total_charge TotalChargePct)
(:map net_charge NetChargePct)
(:map hydrophobics HydrophobicsPct)
(:map pct_A AlaPct)
(:map pct_R ArgPct)
(:map pct_N AsnPct)
(:map pct_D AspPct)

```



```
(:map pct_C CysPct)
(:map pct_E GluPct)
(:map pct_Q GlnPct)
(:map pct_G GlyPct)
(:map pct_H HisPct)
(:map pct_I IlePct)
(:map pct_L LeuPct)
(:map pct_K LysPct)
(:map pct_M MetPct)
(:map pct_F PhePct)
(:map pct_P ProPct)
(:map pct_S SerPct)
(:map pct_T ThrPct)
(:map pct_W TrpPct)
(:map pct_Y TyrPct)
(:map pct_V ValPct))
((:qry SAPSQueryable
()
()))))
```

```
(:src Phobius
(:cls TransmembraneRegion
(:trg tm)
()
((:map id #ID)
(:map src #SOURCE)
(:map desc Desc)
(:map start Start)
(:map stop Stop))
((:qry PhobiusQueryable
()
()))))
(:cls SignalPeptide
(:trg sp)
()
((:map id #ID)
(:map src #SOURCE)
(:map desc Desc)
(:map start Start)
(:map stop Stop))
((:qry PhobiusQueryable
()
()))))
```

())))

```
(:src GenNavGO
(:cls HierarchicalTerm
(:trg term)
((:prp Type .))
((:map id #ID)
 (:map src #SOURCE)
 (:map name Name)
 (:map type Type)
 (:map parents Parents))
((:qry GenNavGOIDQueryable
())
()))))
```

```
(:src AmiGO
(:cls HierarchicalTerm
(:trg term)
((:prp Type .))
((:map id #ID)
 (:map src #SOURCE)
 (:map name Name)
 (:map type Type))
((:qry AmiGOIDQueryable
())
()))))
```

```
(:src XtalPred
(:cls CrystallizabilityAttribute
(:trg xtal)
())
((:map class #ID)
 (:map src #SOURCE))
((:qry XtalPredSequenceQueryable
(:prp Length .)
(:prp Gravy .)
(:prp InstabilityIndex .)
(:prp IsoelectricPoint .)
(:prp CoiledCoils .)
(:prp LongestDisorderRegion .)
(:prp PctCoilStructure .))
```

```

(:prp TransmembraneHelices .)
(:prp SignalPeptides .)
(:prp InsertionsScore .)
(:prp HomologsInNR .)
(:prp HomologsInPDB .))
()))))

(:src UniProt
(:cls Protein
(:trg protein)
((:prp EntryName .)
(:prp Accession .)
(:prp Status .)
(:prp UniProt2InterPro *)
(:prp UniProt2Pfam * )
(:prp UniProt2GO *))
(:map accession #ID)
(:map entryname EntryName)
(:map pname Name)
(:map src #SOURCE)
(:map org Species)
(:map status Status)
(:map interpros UniProt2InterPro)
(:map pfams UniProt2Pfam)
(:map gos UniProt2GO)
(:map sequence Sequence))
(:qry UniProtIDQueryable
()
())
(:qry UniProtSequenceQueryable
((:prp Length .)
(:prp Score .)
(:prp Evaluate .)
(:prp Identity .))
((:par E-value-min 1e-35))))))

(:src CDD
(:cls Domain
(:trg domain)
((:prp Accession .))
(:map ext_id #ID) ; Using external id as source id cdd\d+, COG\d+, etc...
(:map src #SOURCE)

```

```

(:map name Name)
(:map cdd_id Accession)) ; Using PSSM-id as secondary
((:qry CDDSequenceQueryable
(:prp Evaluate .)
(:prp Identity .)
(:prp Length .)
(:prp Score .))
((:par E-value-min 1e-10))))))

```

```

(:src PDB
(:cls Structure
(:trg structure)
(:prp Function .)
(:prp PDB2GO *)
(:prp ExperimentalMethod .)
(:prp Type .)
(:prp Classification .)
(:prp Organism *))
((:map id #ID)
(:map src #SOURCE)
(:map name Name)
(:map type Type)
(:map classification Classification)
(:map organisms Organism)
(:map pdb2go PDB2GO))
((:qry PDBIDQueryable
()
())
(:qry PDBSequenceQueryable
(:prp Evaluate .))
()))))

```

```

(:src NCBI
(:cls Protein
(:trg protein)
(:prp Accession .)
(:prp NCBI2EntrezGene *))
((:map id #ID)
(:map src #SOURCE)
(:map name Name)
(:map species Species)

```

```

(:map gi Accession)
(:map sequence Sequence)
(:map geneid NCBI2EntrezGene))
((:qry ProteinSequenceQueryable
(:prp Evaluate .)
(:prp Identity .)
(:prp Length .)
(:prp Score .))
((:par E-value-min 1e-35))))))

(:src EntrezGene
(:cls Gene
(:trg gene)
((:prp EntrezGene2GO *)
(:prp EntrezGene2UniProt *))
(:map id #ID)
(:map src #SOURCE)
(:map tax Species)
(:map entrezgene2go EntrezGene2GO)
(:map entrezgene2uniprot EntrezGene2UniProt))
(:qry EntrezGeneIDQueryable
()
()))))

(:src InterPro
(:cls Domain
(:trg iptype)
((:prp InterPro2EC *)
(:prp InterPro2PDB *)
(:prp InterPro2GO *)
(:prp InterPro2CATH *)
(:prp InterPro2SCOP *)
(:prp Taxonomy *))
(:map id #ID)
(:map src #SOURCE)
(:map desc Name)
(:map ec InterPro2EC)
(:map pdb InterPro2PDB)
(:map go InterPro2GO)
(:map cath InterPro2CATH)
(:map scop InterPro2SCOP)
(:map tax Taxonomy))

```

```

((:qry InterProIDQueryable
()
()))
(:qry InterProSequenceQueryable
((:prp Evaluate .)
(:prp Start .)
(:prp Stop .))
()))
(:cls Family
(:trg Family)
((:prp InterPro2EC *)
(:prp InterPro2PDB *)
(:prp InterPro2GO *)
(:prp InterPro2CATH *)
(:prp InterPro2SCOP *)
(:prp Taxonomy *))
(:map id #ID)
(:map src #SOURCE)
(:map desc Name)
(:map ec InterPro2EC)
(:map pdb InterPro2PDB)
(:map go InterPro2GO)
(:map cath InterPro2CATH)
(:map scop InterPro2SCOP)
(:map tax Taxonomy))
(:qry InterProIDQueryable
()
()))
(:qry InterProSequenceQueryable
((:prp Evaluate .)
(:prp Start .)
(:prp Stop .))
()))

;;;;;;;;;;;;;;;;;;;;;;;;; Link declarations ;;;;;;;;;;;;;;;;;;;;;;;;;;
; (:lnk <name>
; (:gen <generator>)
; (:src <name> (:cls <name> (:prp <name>)))
; (:src <name> (:cls <name> (:qry <name>
; ( {(:map <label> <name>)} ))))

```

```

;
; Seed links
;
(:lnk Seed2NCBI_Protein_Link
(:gen NCBIProteinSequenceSearchGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src NCBI (:cls Protein (:qry ProteinSequenceQueryable
((:map evaluate Evaluate)
(:map identity Identity)
(:map length Length)
(:map score Score))))))

(:lnk Seed2UniProt_Protein_Link
(:gen UniProtSequenceGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src UniProt (:cls Protein (:qry UniProtSequenceQueryable
((:map len Length)
(:map score Score)
(:map evaluate Evaluate)
(:map identity Identity))))))

(:lnk Seed2Superfamily_Superfamily_Link
(:gen SuperfamilySequenceGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src Superfamily (:cls SuperfamilyCls (:qry SuperfamilySequenceQueryable
((:map evaluate Evaluate))))))

(:lnk Seed2TIGRFAM_Family_Link
(:gen TIGRFAMSequenceGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src TIGRFAM (:cls Family (:qry TIGRFAMQueryable
((:map score Score)
(:map evaluate Evaluate))))))

(:lnk Seed2XtalPred_Crystallizability_Link
(:gen XtalPredSequenceGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src XtalPred (:cls CrystallizabilityAttribute (:qry XtalPredSequenceQueryable
((:map len Length)
(:map gravity Gravity)
(:map ii InstabilityIndex)
(:map iso IsoelectricPoint)

```

```

(:map cc CoiledCoils)
(:map ldr LongestDisorderRegion)
(:map pcs PctCoilStructure)
(:map tms TransmembraneHelices)
(:map sp SignalPeptides)
(:map is InsertionsScore)
(:map hnr HomologsInNR)
(:map hpdb HomologsInPDB))))))

(:lnk Seed2PDB_Structure_Link
(:gen PDBSequenceGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src PDB (:cls Structure (:qry PDBSequenceQueryable
((:map eval Evaluate))))))

(:lnk Seed2BioCyc_Protein_Link
(:gen BioCycSequenceGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src BioCyc (:cls Protein (:qry BioCycSequenceQueryable
((:map evaluate Evaluate)
(:map score Score)
(:map length Length))))))

; Note the query also will fire for SignalPeptide -- the trigger in the generator
; tells the engine what class to instantiate, and it is not enforced by the link.
(:lnk Seed2Phobius_Link
(:gen PhobiusGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src Phobius (:cls TransmembraneRegion (:qry PhobiusQueryable ())))))

(:lnk Seed2SAPS_Link
(:gen SAPSSequenceGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src SAPS (:cls SequenceComposition (:qry SAPSQueryable ())))))

(:lnk Seed2InterProSequence_Link
(:gen InterProSequenceGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src InterPro (:cls Domain (:qry InterProSequenceQueryable
((:map eval Evaluate)
(:map start Start)
(:map stop Stop))))))

```



```

(:lnk Seed2CDD_Domain_Link
(:gen CDDProteinSequenceSearchGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src CDD (:cls Domain (:qry CDDSequenceQueryable
((:map evaluate Evaluate)
(:map identity Identity)
(:map length Length)
(:map score Score))))))

(:lnk Seed2KEGG_Gene_Link
(:gen KEGGSequenceGenerator)
(:src Seed (:cls ProteinSequenceQuery (:prp #ID)))
(:src KEGG (:cls Gene (:qry KEGGSequenceQueryable
((:map eval Evaluate)
(:map bits Bits))))))

;
; BioCyc links
;
(:lnk BioCyc_Protein2BioCyc_Pathway
(:gen BioCycPathwayGenerator)
(:src BioCyc (:cls Protein (:prp BioCyc2Pathway)))
(:src BioCyc (:cls Pathway (:qry BioCycPathwayQueryable ())))))

(:lnk BioCyc_Protein2AmiGO_Link
(:gen GOAmiGOIDGenerator)
(:src BioCyc (:cls Protein (:prp BioCyc2GO)))
(:src AmiGO (:cls HierarchicalTerm (:qry AmiGOIDQueryable ())))))

;lnk was omitted originally (whoops)
(:lnk BioCyc_Protein2GenNavGO_Link
(:gen GOGenNavIDGenerator)
(:src BioCyc (:cls Protein (:prp BioCyc2GO)))
(:src GenNavGO (:cls HierarchicalTerm (:qry GenNavGOIDQueryable()))))

;
; PDB Links
;
(:lnk PDB_Structure2GenNavGO_Link
(:gen GOGenNavIDGenerator)
(:src PDB (:cls Structure (:prp PDB2GO)))

```

```

(:src GenNavGO (:cls HierarchicalTerm (:qry GenNavGOIDQueryable ())))

; Use AmiGO as backup
(:lnk PDB_Structure2AmiGO_Link
(:gen GOAmiGOIDGenerator)
(:src PDB (:cls Structure (:prp PDB2GO)))
(:src AmiGO (:cls HierarchicalTerm (:qry AmiGOIDQueryable ())))

;
; InterPro links
;
(:lnk InterPro_Domain2GenNavGO_Link
(:gen GOGenNavIDGenerator)
(:src InterPro (:cls Domain (:prp InterPro2GO)))
(:src GenNavGO (:cls HierarchicalTerm (:qry GenNavGOIDQueryable ())))

(:lnk InterPro_Family2GenNavGO_Link
(:gen GOGenNavIDGenerator)
(:src InterPro (:cls Family (:prp InterPro2GO)))
(:src GenNavGO (:cls HierarchicalTerm (:qry GenNavGOIDQueryable ())))

; Use AmiGO as backup
(:lnk InterPro_Domain2AmiGO_Link
(:gen GOAmiGOIDGenerator)
(:src InterPro (:cls Domain (:prp InterPro2GO)))
(:src AmiGO (:cls HierarchicalTerm (:qry AmiGOIDQueryable ())))

(:lnk InterPro_Family2AmiGO_Link
(:gen GOAmiGOIDGenerator)
(:src InterPro (:cls Family (:prp InterPro2GO)))
(:src AmiGO (:cls HierarchicalTerm (:qry AmiGOIDQueryable ())))

;
; TIGRFAM links
;
(:lnk TIGRFAM_Family2AmiGO_Link
(:gen GOAmiGOIDGenerator)
(:src TIGRFAM (:cls Family (:prp TIGRFAM2GO)))
(:src AmiGO (:cls HierarchicalTerm (:qry AmiGOIDQueryable ())))

(:lnk TIGRFAM_Family2GenNavGO_Link
(:gen GOGenNavIDGenerator)

```

```

(:src TIGRFAM (:cls Family (:prp TIGRFAM2GO)))
(:src GenNavGO (:cls HierarchicalTerm (:qry GenNavGOIDQueryable ())))

(:lnk TIGRFAM_Family2InterPro_Link
(:gen InterProIDGenerator)
(:src TIGRFAM (:cls Family (:prp TIGRFAM2InterPro)))
(:src InterPro (:cls Family (:qry InterProIDQueryable()))))

;
; UniProt links
;
(:lnk UniProt_Protein2GenNavGO_Link
(:gen GOGenNavIDGenerator)
(:src UniProt (:cls Protein (:prp UniProt2GO)))
(:src GenNavGO (:cls HierarchicalTerm (:qry GenNavGOIDQueryable ())))

(:lnk UniProt_Protein2AmiGO_Link
(:gen GOAmiGOIDGenerator)
(:src UniProt (:cls Protein (:prp UniProt2GO)))
(:src AmiGO (:cls HierarchicalTerm (:qry AmiGOIDQueryable ())))

;
; GenNavGO links
;
(:lnk GenNavGO_HierarchicalTerm2GenNavGO_HierarchicalTerm
(:gen GOGenNavIDGenerator)
(:src GenNavGO (:cls HierarchicalTerm (:prp Parents)))
(:src GenNavGO (:cls HierarchicalTerm (:qry GenNavGOIDQueryable ())))

;
; EntrezGene links
;
(:lnk EntrezGene_Gene2AmiGO_Link
(:gen GOAmiGOIDGenerator)
(:src EntrezGene (:cls Gene (:prp EntrezGene2GO)))
(:src AmiGO (:cls HierarchicalTerm (:qry AmiGOIDQueryable ())))

(:lnk EntrezGene_Gene2GenNavGO_Link
(:gen GOGenNavIDGenerator)
(:src EntrezGene (:cls Gene (:prp EntrezGene2GO)))
(:src GenNavGO (:cls HierarchicalTerm (:qry GenNavGOIDQueryable ())))

```

```

(:lnk EntrezGene_Gene2UniProt_Protein_Link
(:gen UniProtIDGenerator)
(:src EntrezGene (:cls Gene (:prp EntrezGene2UniProt)))
(:src UniProt (:cls Protein (:qry UniProtIDQueryable ())))

;
; NCBI links
;
(:lnk NCBI_Protein2EntrezGene_Gene_Link
(:gen EntrezGeneIDGenerator)
(:src NCBI (:cls Protein (:prp NCBI2EntrezGene)))
(:src EntrezGene (:cls Gene (:qry EntrezGeneIDQueryable ())))

;
; KEGG links
;
(:lnk KEGG_Gene2Term_Link
(:gen KEGGOrthologyGenerator)
(:src KEGG (:cls Gene (:prp Gene2Term)))
(:src KEGG (:cls Term (:qry KEGGOrthologyQueryable ())))

(:lnk KEGG_Gene2Pathway_Link
(:gen KEGGPathwayGenerator)
(:src KEGG (:cls Gene (:prp Gene2Pathway)))
(:src KEGG (:cls Pathway (:qry KEGGPathwayQueryable ())))

(:lnk KEGG_Term2Pathway_Link
(:gen KEGGPathwayGenerator)
(:src KEGG (:cls Term (:prp Term2Pathway)))
(:src KEGG (:cls Pathway (:qry KEGGPathwayQueryable ())))

(:lnk KEGG_Gene2EntrezGene_Gene_Link
(:gen EntrezGeneIDGenerator)
(:src KEGG (:cls Gene (:prp KEGG2EntrezGene)))
(:src EntrezGene (:cls Gene (:qry EntrezGeneIDQueryable ())))

(:lnk KEGG_Gene2Uniprot_Protein_Link
(:gen UniProtIDGenerator)
(:src KEGG (:cls Gene (:prp KEGG2UniProt)))
(:src UniProt (:cls Protein (:qry UniProtIDQueryable ())))
===== END BLOCK =====

```

A.2 DaRQL grammar

```

    query = target_clause, ws, from_clause, ws, restrict_clause;
target_clause = "TARGET", ws, nodelist;
    nodelist = ( node | varset ), [ ',', nodelist ];
    node = entity_node | source_node | domain_node;
entity_node = { alpha };
    varset = var, "'", ( entity_node | source_node | domain_node );
    var = "?", { alpha };
source_node = "(", { alpha }, "'";
domain_node = "@", { alpha };
from_clause = "FROM", ws, nodelist;
restrict_clause = "RESTRICT", ws, constraint_list;
constraint_list = constraint, [ ',', constraint_list ];
    constraint = path_constraint | node_constraint;
node_constraint = varacc, binop, varacc;
    binop = "!=" | "=" | ">=" | "<=" | ">" | "<";
    varacc = var | literal | number | divfunc | evfunc;
divfunc = "divcount", "(", var, "'";
evfunc = "order", "(", var, "'";
path_constraint = "{", path, "'";
    path = pathable, "'", path;
pathable = source_node | domain_node | entity_node | "*" | var | conjunct | disjunct;
conjunct = "(", node, "&", node "'";
disjunct = "(", node, "|", node "'";
    number = [ - ], { digit }, [ ".", number ];
    digit = ? all digits between 0 and 9 inclusive ?;
    alpha = ? all alphabetical characters ?;
literal = "{ alpha }";
ws = { ' ' };

```

Figure A.1: DaRQL production rules, in extended Backus-Naur Form.

A.3 DaRQL query planning benchmarks

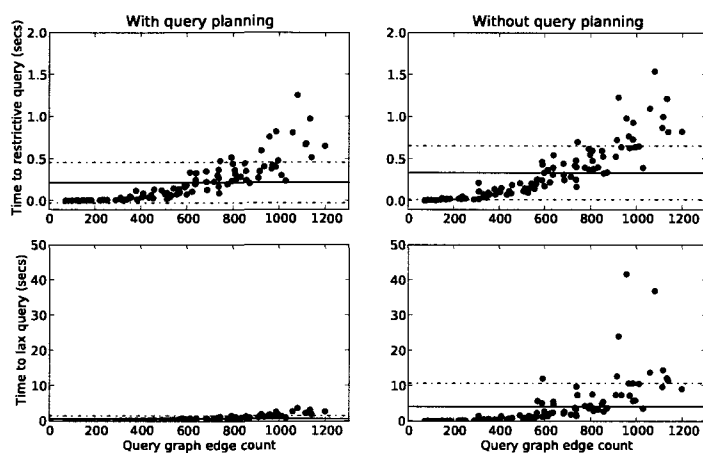


Figure A.2: Query execution times for 99 randomly selected proteins from yeast, without query planning and with query planning for a restrictive query with many constraints and for a loosely-defined query with few constraints; solid horizontal lines mark the means for each query, and dotted horizontal lines the standard deviations. Benchmarks were done on an Intel dual core 2.66GHz Linux machine with 3GB of RAM.

Appendix B
LEARNING EXPERIMENTS

B.1 Characteristics of the query graph

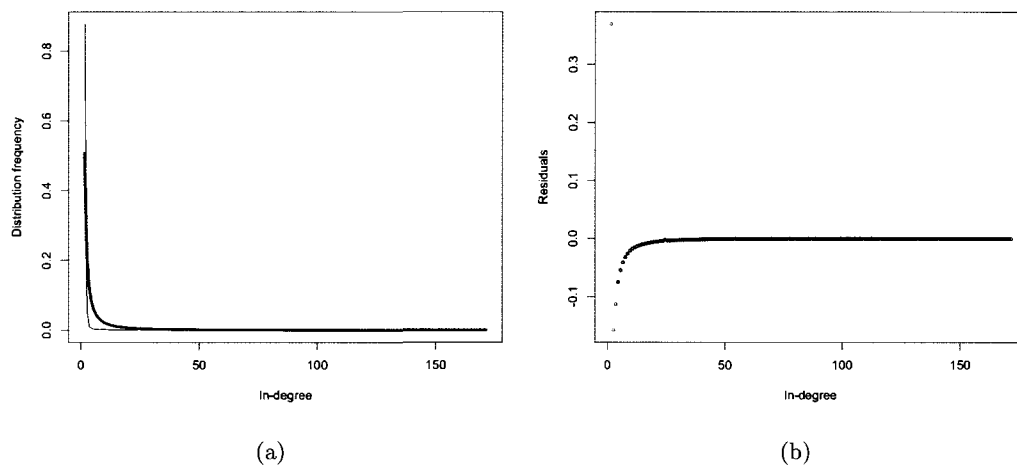


Figure B.1: A power-law model fitted to the distribution frequency of in-degrees in the query graphs (Fig. B.1a), and the residuals (Fig. B.1b). The model was fitted *via* unweighted least-squares regression using the formula $P_k = k^{-\alpha}$, with P_k the distribution frequency for in-degree k . α was estimated to be 1.67.

B.2 Generalized virulence results

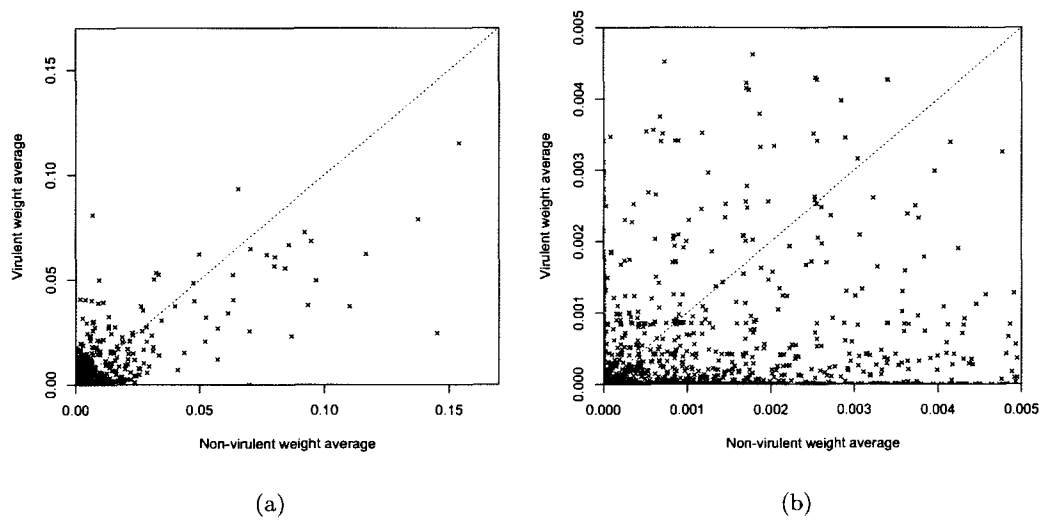


Figure B.2: Individual propagation weights for GO terms, based on virulent and non-virulent records; Fig. B.2b is a zoomed-in subset of Fig. B.2a.

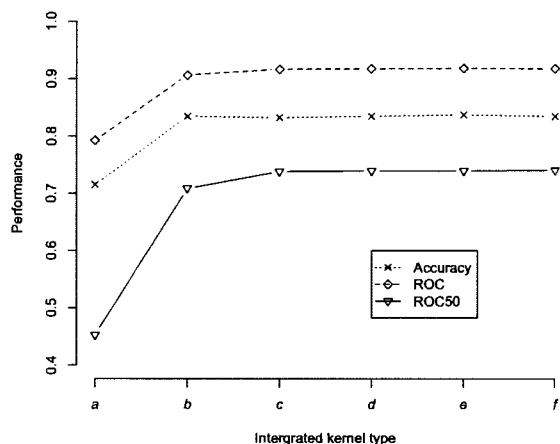


Figure B.3: Progressive results of integrating kernel sources *via* (3.7) into a sequence-baseline (a) SVM classifier (source order is (a+) GENNAV (\rightarrow b), (a + b+) AMIGO (\rightarrow c), INTERPRO (d), KEGG (e), CDD (f)). There was no significant improvement in using integrated kernels using equal weighting; *ad hoc* weighting of the kernels did show some improvement (not shown), but weight-tuning was not done comprehensively.

B.3 Statistical significance of specific virulence

Tables B.1 to B.11 contain the statistical significance tests for the six five-fold cross-validations of the integrated data sources against the baseline, according to the results of paired *t*-tests. Each cell contains a Bonferroni-adjusted p-value if the source or method describing the row statistically outperforms the source or method on the column, a (+) if the source along the row is not statistically better than the source or method along the column and a (-) if the source or method along the row is the same or worse (statistically or otherwise). Significance for these comparisons was set at $\alpha = 0.05$.

For these tables, the sources are listed as BioCyc (BCyc), CDD, GenNav (GN), AmiGO (AG), InterPro (IPro), KEGG, TIGRFAM (TFAM), PDB, BLAST with *k*NN (kNNB), 3mers frequencies and BLAST with SVM (BSVM).

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	-	+	-	-	-	-
CDD	3.79e-13	-	-	-	-	4.09e-07	1.22e-14	1.87e-12	+	+	1.14e-04
GN	0.00e+00	4.95e-12	-	+	+	0.00e+00	0.00e+00	0.00e+00	5.13e-10	1.29e-11	3.04e-12
AG	0.00e+00	5.75e-12	-	-	+	0.00e+00	0.00e+00	0.00e+00	3.67e-10	1.09e-11	2.04e-12
IPro	0.00e+00	6.23e-13	-	-	-	0.00e+00	0.00e+00	0.00e+00	4.07e-11	9.72e-12	6.84e-13
KEGG	1.49e-02	-	-	-	-	-	2.03e-05	+	-	-	-
TFAM	-	-	-	-	-	-	-	-	-	-	-
PDB	6.26e-03	-	-	-	-	-	3.30e-05	-	-	-	-
kNNB	6.65e-06	-	-	-	-	+	5.83e-06	3.28e-02	-	-	+
3mer	6.66e-09	-	-	-	-	1.84e-03	4.34e-09	8.55e-06	+	-	4.42e-04
BSVM	+	-	-	-	-	+	+	+	-	-	-

Table B.1: ROC p-values for the Adherence virulent class.

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	-	-	-	-	-	-
CDD	4.96e-06	-	-	-	-	3.05e-02	3.19e-02	+	-	-	-
GN	1.73e-12	9.53e-07	-	+	7.99e-04	1.59e-13	1.11e-12	5.23e-09	6.04e-06	-	3.89e-05
AG	5.86e-13	1.25e-06	-	-	5.74e-04	2.44e-14	1.10e-12	9.75e-09	5.83e-07	-	2.74e-05
IPro	7.58e-12	+	-	-	-	5.99e-09	2.33e-09	2.51e-03	+	-	+
KEGG	+	-	-	-	-	-	+	-	-	-	-
TFAM	+	-	-	-	-	-	-	-	-	-	-
PDB	2.28e-04	-	-	-	-	+	+	-	-	-	-
kNNB	3.21e-04	+	-	-	-	3.36e-02	1.55e-03	+	-	-	-
3mer	1.22e-14	6.36e-08	+	+	2.57e-06	0.00e+00	1.22e-14	1.11e-11	1.25e-06	-	3.83e-07
BSVM	2.91e-05	+	-	-	-	1.50e-04	5.49e-05	+	+	-	-

Table B.2: ROC p-values for the Surface factor virulent class.

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	-	-	-	-	-	+
CDD	2.43e-10	-	-	-	-	-	1.09e-08	1.36e-03	+	+	5.47e-09
GN	0.00e+00	4.15e-07	-	1.96e-05	5.61e-06	1.76e-06	8.55e-14	8.74e-11	1.35e-09	1.58e-10	7.33e-14
AG	0.00e+00	2.67e-05	-	-	7.01e-04	1.84e-04	1.98e-12	4.57e-09	1.18e-06	5.46e-09	1.47e-13
IPro	1.03e-12	+	-	-	-	+	3.18e-13	1.41e-05	+	1.97e-03	6.81e-12
KEGG	9.71e-08	+	-	-	-	-	4.29e-07	6.61e-03	+	2.31e-03	1.18e-10
TFAM	+	-	-	-	-	-	-	-	-	-	+
PDB	1.74e-05	-	-	-	-	-	6.42e-04	-	-	-	2.19e-05
kNNB	1.31e-04	-	-	-	-	-	6.04e-04	+	-	+	2.73e-05
3mer	6.02e-03	-	-	-	-	-	4.93e-02	+	-	-	6.51e-05
BSVM	-	-	-	-	-	-	-	-	-	-	-

Table B.3: ROC p-values for the Invasion virulent class.

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	-	1.22e-06	-	-	4.25e-08	4.45e-08
CDD	1.65e-11	-	-	-	+	3.15e-09	4.03e-13	5.06e-08	1.29e-02	2.81e-13	0.00e+00
GN	0.00e+00	1.51e-06	-	2.98e-05	1.67e-06	0.00e+00	0.00e+00	2.12e-11	3.18e-07	0.00e+00	0.00e+00
AG	7.94e-13	+	-	-	1.43e-02	7.33e-14	0.00e+00	1.78e-08	1.33e-03	0.00e+00	0.00e+00
IPro	2.21e-08	-	-	-	-	5.18e-07	4.88e-14	2.50e-04	+	1.40e-12	9.77e-14
KEGG	+	-	-	-	-	-	3.81e-05	-	-	1.04e-10	4.00e-09
TFAM	-	-	-	-	-	-	-	-	-	+	+
PDB	5.13e-03	-	-	-	-	+	2.32e-10	-	-	2.86e-10	1.48e-12
kNNB	+	-	-	-	-	+	2.83e-06	+	-	1.58e-07	1.10e-07
3mer	-	-	-	-	-	-	-	-	-	-	-
BSVM	-	-	-	-	-	-	-	-	-	+	-

Table B.4: ROC p-values for the Transport and uptake virulent class.

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	-	3.90e-06	-	-	-	+
CDD	1.83e-13	-	-	-	-	5.10e-06	0.00e+00	5.79e-06	4.59e-05	9.97e-07	3.68e-12
GN	0.00e+00	6.85e-12	-	1.22e-04	5.84e-07	0.00e+00	0.00e+00	0.00e+00	1.95e-13	0.00e+00	0.00e+00
AG	0.00e+00	1.35e-09	-	-	8.43e-04	1.10e-13	0.00e+00	9.77e-14	1.48e-12	3.66e-14	0.00e+00
IPro	0.00e+00	6.86e-05	-	-	-	2.09e-11	0.00e+00	2.31e-12	3.40e-10	1.20e-10	0.00e+00
KEGG	1.73e-05	-	-	-	-	-	2.52e-10	-	+	+	4.62e-05
TFAM	-	-	-	-	-	-	-	-	-	-	-
PDB	1.18e-05	-	-	-	-	+	8.18e-13	-	+	+	4.17e-09
kNNB	6.85e-03	-	-	-	-	-	4.82e-07	-	-	+	1.70e-03
3mer	+	-	-	-	-	-	3.82e-03	-	-	-	3.02e-02
BSVM	-	-	-	-	-	-	+	-	-	-	-

Table B.5: ROC p-values for the Toxin virulent class.

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	-	-	-	-	-	-
CDD	0.00e+00	-	-	-	-	6.72e-13	2.44e-14	6.36e-03	+	+	1.15e-05
GN	0.00e+00	2.84e-05	-	+	8.01e-05	0.00e+00	0.00e+00	3.56e-09	1.55e-07	6.09e-07	4.90e-09
AG	0.00e+00	1.94e-05	-	-	1.38e-04	0.00e+00	0.00e+00	6.41e-09	1.84e-07	4.61e-07	1.07e-08
IPro	0.00e+00	+	-	-	-	3.66e-14	0.00e+00	1.04e-05	+	+	4.12e-06
KEGG	1.60e-02	-	-	-	-	-	+	-	-	-	-
TFAM	+	-	-	-	-	-	-	-	-	-	-
PDB	8.67e-10	-	-	-	-	2.44e-05	2.87e-08	-	-	-	+
kNNB	3.05e-13	-	-	-	-	3.20e-10	1.38e-11	+	-	-	4.03e-02
3mer	2.44e-14	-	-	-	-	1.92e-12	3.13e-12	+	+	-	5.56e-04
BSVM	2.97e-05	-	-	-	-	4.12e-02	4.88e-04	-	-	-	-

Table B.6: ROC p-values for the Catalysis virulent class.

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	-	-	-	-	-	3.54e-02
CDD	0.00e+00	-	-	-	-	-	6.40e-11	1.22e-14	+	1.02e-06	4.27e-13
GN	0.00e+00	0.00e+00	-	3.90e-06	1.02e-07	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
AG	0.00e+00	0.00e+00	-	-	+	1.44e-11	0.00e+00	0.00e+00	1.71e-13	0.00e+00	0.00e+00
IPro	0.00e+00	0.00e+00	-	-	-	5.00e-11	0.00e+00	0.00e+00	1.53e-12	0.00e+00	0.00e+00
KEGG	0.00e+00	2.74e-08	-	-	-	-	0.00e+00	0.00e+00	3.30e-09	4.15e-13	0.00e+00
TFAM	1.94e-06	-	-	-	-	-	-	5.87e-04	-	+	3.70e-07
PDB	+	-	-	-	-	-	-	-	-	-	1.67e-04
kNNB	4.17e-07	-	-	-	-	-	+	1.93e-05	-	+	2.32e-08
3mer	2.18e-02	-	-	-	-	-	-	+	-	-	2.81e-05
BSVM	-	-	-	-	-	-	-	-	-	-	-

Table B.7: ROC p-values for the Secretion virulent class.

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	-	5.85e-05	1.75e-10	+	1.74e-06	4.66e-09
CDD	1.46e-08	-	-	-	-	-	3.66e-14	0.00e+00	3.63e-04	1.29e-12	0.00e+00
GN	0.00e+00	6.11e-14	-	+	1.55e-10	6.28e-10	0.00e+00	0.00e+00	3.66e-14	0.00e+00	0.00e+00
AG	0.00e+00	8.55e-14	-	-	2.65e-10	2.66e-09	0.00e+00	0.00e+00	1.83e-13	0.00e+00	0.00e+00
IPro	1.99e-12	8.77e-05	-	-	-	+	0.00e+00	0.00e+00	6.66e-07	0.00e+00	0.00e+00
KEGG	3.71e-08	+	-	-	-	-	2.09e-12	0.00e+00	4.77e-08	1.71e-13	3.66e-14
TFAM	-	-	-	-	-	-	-	1.48e-05	-	+	1.15e-05
PDB	-	-	-	-	-	-	-	-	-	-	+
kNNB	-	-	-	-	-	-	+	2.61e-08	-	4.04e-05	1.24e-05
3mer	-	-	-	-	-	-	-	+	-	-	+
BSVM	-	-	-	-	-	-	-	-	-	-	-

Table B.8: ROC p-values for the Motility virulent class.

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	1.40e-08	2.56e-06	6.64e-03	+	3.63e-09	4.98e-11
CDD	8.12e-11	-	-	-	-	0.00e+00	0.00e+00	0.00e+00	2.80e-05	1.10e-13	0.00e+00
GN	0.00e+00	1.24e-06	-	1.22e-14	-	0.00e+00	0.00e+00	0.00e+00	1.22e-14	0.00e+00	0.00e+00
AG	1.16e-08	+	-	-	-	3.66e-14	1.47e-13	2.98e-10	3.19e-08	0.00e+00	0.00e+00
IPro	0.00e+00	2.66e-12	1.67e-02	9.43e-12	-	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
KEGG	-	-	-	-	-	-	-	-	-	+	+
TFAM	-	-	-	-	-	9.74e-06	-	-	-	4.34e-05	8.13e-07
PDB	-	-	-	-	-	5.32e-06	+	-	-	9.41e-04	7.37e-06
kNNB	-	-	-	-	-	5.62e-07	+	+	-	5.01e-07	2.32e-06
3mer	-	-	-	-	-	-	-	-	-	-	+
BSVM	-	-	-	-	-	-	-	-	-	-	-

Table B.9: ROC p-values for the Antibiotic resistance virulent class.

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	-	2.35e-03	-	-	-	+
CDD	1.22e-14	-	-	-	-	3.98e-03	0.00e+00	2.54e-12	7.92e-04	7.12e-08	2.03e-11
GN	0.00e+00	1.38e-03	-	+	+	1.95e-13	0.00e+00	2.32e-13	2.34e-09	1.05e-12	1.15e-11
AG	1.22e-14	6.70e-04	-	-	-	2.22e-11	0.00e+00	2.08e-13	2.88e-08	5.25e-13	1.39e-11
IPro	0.00e+00	1.66e-07	-	+	-	1.56e-11	0.00e+00	0.00e+00	2.37e-12	3.66e-14	0.00e+00
KEGG	4.77e-05	-	-	-	-	-	1.39e-08	9.58e-04	+	6.99e-03	7.39e-04
TFAM	-	-	-	-	-	-	-	-	-	-	-
PDB	+	-	-	-	-	-	2.25e-03	-	-	-	+
kNNB	2.90e-03	-	-	-	-	-	1.86e-05	7.71e-03	-	1.76e-02	5.68e-03
3mer	+	-	-	-	-	-	4.50e-02	+	-	-	+
BSVM	-	-	-	-	-	-	+	-	-	-	-

Table B.10: ROC p-values for the Defense virulent class.

	BCyc	CDD	GN	AG	IPro	KEGG	TFAM	PDB	kNNB	3mer	BSVM
BioC	-	-	-	-	-	+	1.57e-04	+	+	-	1.68e-08
CDD	2.20e-13	-	-	-	-	3.53e-08	1.22e-14	2.44e-14	2.58e-06	3.87e-05	3.66e-14
GN	0.00e+00	1.47e-04	-	+	7.28e-06	0.00e+00	1.22e-14	0.00e+00	4.80e-12	4.88e-14	0.00e+00
AG	8.55e-14	6.86e-03	-	-	8.57e-03	1.22e-14	1.34e-13	9.77e-14	1.59e-11	1.76e-12	0.00e+00
IPro	8.55e-14	+	-	-	-	5.62e-12	3.54e-13	3.42e-13	7.67e-10	9.05e-09	3.66e-14
KEGG	-	-	-	-	-	-	+	+	-	-	4.38e-06
TFAM	-	-	-	-	-	-	-	-	-	-	1.93e-04
PDB	-	-	-	-	-	-	+	-	-	-	1.46e-04
kNNB	-	-	-	-	-	+	+	+	-	-	2.19e-04
3mer	+	-	-	-	-	+	2.06e-02	+	+	-	1.96e-10
BSVM	-	-	-	-	-	-	-	-	-	-	-

Table B.11: ROC p-values for the Other virulent class.

	1	2	3	4	5	6	7	8	9	10	11
BioC	0.647	0.582	0.625	0.733	0.648	0.626	0.669	0.833	0.737	0.652	0.715
CDD	0.716	0.669	0.693	0.798	0.731	0.828	0.748	0.883	0.796	0.718	0.785
GN	0.793	0.801	0.766	0.842	0.808	0.900	0.850	0.967	0.851	0.751	0.831
AG	0.791	0.796	0.746	0.818	0.793	0.898	0.834	0.967	0.801	0.749	0.822
IPro	0.785	0.720	0.707	0.795	0.763	0.850	0.818	0.909	0.874	0.751	0.797
KEGG	0.666	0.607	0.703	0.737	0.684	0.664	0.778	0.905	0.668	0.688	0.700
TFAM	0.637	0.595	0.626	0.683	0.616	0.631	0.707	0.789	0.710	0.633	0.684
PDB	0.664	0.647	0.663	0.757	0.689	0.774	0.685	0.730	0.712	0.656	0.694
kNNB	0.695	0.679	0.678	0.766	0.678	0.803	0.726	0.827	0.737	0.681	0.706
3mer	0.708	0.813	0.663	0.653	0.661	0.818	0.698	0.763	0.663	0.657	0.722
BSVM	0.668	0.690	0.611	0.661	0.628	0.727	0.642	0.720	0.654	0.647	0.628

Table B.12: ROC means across specific virulence classes, derived from six five-fold cross-validations (corresponding to the significance tests).

	1	2	3	4	5	6	7	8	9	10	11
BioC	0.759	0.727	0.732	0.841	0.849	0.829	0.868	0.935	0.700	0.661	0.791
CDD	0.749	0.900	0.744	0.868	0.836	0.952	0.856	0.919	0.818	0.719	0.793
GN	0.741	0.637	0.657	0.706	0.708	0.726	0.741	0.949	0.735	0.711	0.577
AG	0.743	0.589	0.672	0.653	0.751	0.725	0.745	0.947	0.699	0.623	0.654
IPro	0.690	0.775	0.678	0.767	0.651	0.875	0.775	0.926	0.740	0.621	0.598
KEGG	0.731	0.568	0.715	0.779	0.741	0.910	0.873	0.955	0.618	0.686	0.666
TFAM	0.845	0.758	0.738	0.820	0.858	0.834	0.917	0.957	0.728	0.727	0.826
PDB	0.827	0.980	0.836	0.915	0.870	0.927	0.923	0.970	0.730	0.817	0.850
kNNB	0.616	0.768	0.667	0.719	0.642	0.764	0.655	0.735	0.729	0.583	0.569
3mer	0.594	0.568	0.585	0.702	0.625	0.738	0.530	0.706	0.595	0.592	0.593
BSVM	0.679	0.788	0.730	0.828	0.711	0.870	0.753	0.842	0.762	0.682	0.755

Table B.13: ROC₅₀ means across specific virulence classes, derived from six five-fold cross-validations.

BIBLIOGRAPHY

- [1] Jmol: an open-source Java viewer for chemical structures in 3d, 2009.
- [2] Ramasubramanian Sundaramoorthy, Paul K Fyfe, and William N Hunter. Structure of staphylococcus aureus esxa suggests a contribution to virulence by action as a transport chaperone and/or adaptor protein. *J Mol Biol*, 383(3):603–14, Nov 2008.
- [3] Healthmap, August 2009.
- [4] Lawrence C Madoff. Promed-mail: an early warning system for emerging diseases. *Clin Infect Dis*, 39(2):227–32, Jul 2004.
- [5] ProMED, August 2009.
- [6] Eurosurveillance, August 2009.
- [7] Kate E Jones, Nikkita G Patel, Marc A Levy, Adam Storeygard, Deborah Balk, John L Gittleman, and Peter Daszak. Global trends in emerging infectious diseases. *Nature*, 451(7181):990–3, Feb 2008.
- [8] A D Baxevanis. The molecular biology database collection: an updated compilation of biological database resources. *Nucleic Acids Res*, 29(1):1–10, Jan 2001.
- [9] Andreas D Baxevanis. The molecular biology database collection: 2002 update. *Nucleic Acids Res*, 30(1):1–12, Jan 2002.
- [10] Andreas D Baxevanis. The molecular biology database collection: 2003 update. *Nucleic Acids Res*, 31(1):1–12, Jan 2003.
- [11] Michael Y Galperin. The molecular biology database collection: 2004 update. *Nucleic Acids Res*, 32(Database issue):D3–22, Jan 2004.
- [12] Michael Y Galperin. The molecular biology database collection: 2005 update. *Nucleic Acids Res*, 33(Database issue):D5–24, Jan 2005.
- [13] Michael Y Galperin. The molecular biology database collection: 2006 update. *Nucleic Acids Res*, 34(Database issue):D3–5, Jan 2006.
- [14] Michael Y Galperin. The molecular biology database collection: 2007 update. *Nucleic Acids Res*, 35(Database issue):D3–4, Jan 2007.
- [15] Michael Y Galperin. The molecular biology database collection: 2008 update. *Nucleic Acids Res*, 36(Database issue):D2–4, Jan 2008.
- [16] Michael Y Galperin and Guy R Cochrane. Nucleic acids research annual database issue and the nar online molecular biology database collection in 2009. *Nucleic Acids Res*, 37(Database issue):D1–4, Jan 2009.
- [17] Gert R. G. Lanckriet, Minghua Deng, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Pacific Symposium on Biocomputing*, pages 300–311, 2004.

- [18] Hon Nian Chua, Wing-Kin Sung, and Limsoon Wong. An efficient strategy for extensive integration of diverse biological data for protein function prediction. *Bioinformatics*, 23(24):3364–3373, 2007.
- [19] Eithon Cadag, Brenton Louie, Peter J. Myler, and Peter Tarczy-Hornoch. BioMediator data integration and inference for functional annotation of anonymous sequences. In Altman et al. [264], pages 343–354.
- [20] Landon Detwiler, Wolfgang Gatterbauer, Brenton Louie, Dan Suci, and Peter Tarczy-Hornoch. Integrating and ranking uncertain scientific data. In *ICDE* [216], pages 1235–1238.
- [21] Terry H Shen, Christopher S Carlson, and Peter Tarczy-Hornoch. Evaluating the accuracy of a functional snp annotation system. *BMC Bioinformatics*, 10 Suppl 9:S11, 2009.
- [22] T M Wassenaar and W Gaastra. Bacterial virulence: can we draw the line? *FEMS Microbiol Lett*, 201(1):1–7, 2001 Jul 10.
- [23] Lihong Chen, Jian Yang, Jun Yu, Zhijian Yao, Lilian Sun, Yan Shen, and Qi Jin. Vfdb: a reference database for bacterial virulence factors. *Nucleic Acids Res*, 33(Database issue):D325–8, Jan 2005.
- [24] Joy Scaria, Umamaheswaran Chandramouli, and Sanjay Kumar Verma. Antibiotic resistance genes online (argo): a database on vancomycin and beta-lactam resistance genes. *Bioinformatics*, 1(1):5–7, 2005.
- [25] Rainer Winnenburg, Thomas K Baldwin, Martin Urban, Chris Rawlings, Jacob Köhler, and Kim E Hammond-Kosack. Phi-base: a new database for pathogen host interactions. *Nucleic Acids Res*, 34(Database issue):D459–64, Jan 2006.
- [26] Leslie Klis McNeil, Claudia Reich, Ramy K Aziz, Daniela Bartels, Matthew Cohoon, Terry Disz, Robert A Edwards, Svetlana Gerdes, Kaitlyn Hwang, Michael Kubal, Gohar Rem Margaryan, Folker Meyer, William Mihalo, Gary J Olsen, Robert Olson, Andrei Osterman, Daniel Paarmann, Tobias Paczian, Bruce Parrello, Gordon D Pusch, Dmitry A Rodionov, Xinghua Shi, Olga Vassieva, Veronika Vonstein, Olga Zagnitko, Fangfang Xia, Jenifer Zinner, Ross Overbeek, and Rick Stevens. The national microbial pathogen database resource (nmpdr): a genomics platform based on subsystem annotation. *Nucleic Acids Res*, 35(Database issue):D347–53, Jan 2007.
- [27] C E Zhou, J Smith, M Lam, A Zemla, M D Dyer, and T Slezak. Mvirdb—a microbial database of protein toxins, virulence factors and antibiotic resistance genes for bio-defence applications. *Nucleic Acids Res*, 35(Database issue):D391–4, Jan 2007.
- [28] Jeremy D Selengut, Daniel H Haft, Tanja Davidsen, Anurhada Ganapathy, Michelle Gwinn-Giglio, William C Nelson, Alexander R Richter, and Owen White. Tigrfams and genome properties: tools for the assignment of molecular function and biological process in prokaryotic genomes. *Nucleic Acids Res*, 35(Database issue):D260–4, Jan 2007.
- [29] In Seok Yang, Chunsun Ryu, Ki Joon Cho, Jin Kwang Kim, Swee Hoe Ong, Wayne P Mitchell, Bong Su Kim, Hee-Bok Oh, and Kyung Hyun Kim. Idbd: infectious disease biomarker database. *Nucleic Acids Res*, 36(Database issue):D455–60, Jan 2008.
- [30] C Nelson Hayes, Diego Diez, Nicolas Joannin, Wataru Honda, Minoru Kanehisa, Mats Wahlgren, Craig E Wheelock, and Susumu Goto. vardb: a pathogen-specific sequence database of protein families involved in antigenic variation. *Bioinformatics*, 24(21):2564–5, Nov 2008.
- [31] Bo Liu and Mihai Pop. Ardb—antibiotic resistance genes database. *Nucleic Acids Res*, 37(Database issue):D443–7, Jan 2009.

- [32] E A Worthey and P J Myler. Protozoan genomes: gene identification and annotation. *Int J Parasitol*, 35(5):495–512, Apr 2005.
- [33] Sohrab P Shah, Yong Huang, Tao Xu, Macaire M S Yuen, John Ling, and B F Francis Ouellette. Atlas - a data warehouse for integrative bioinformatics. *BMC Bioinformatics*, 6:34, 2005.
- [34] Sarah Cohen-Boulakia, Olivier Biton, Susan Davidson, and Christine Froidevaux. BioGuideSRS: querying multiple sources with a user-centric perspective. *Bioinformatics*, 23(10):1301–1303, 2007 May 15.
- [35] Loren Donelson, Peter Tarczy-Hornoch, Peter Mork, Cindy Dolan, Joyce A Mitchell, M Barrier, and Hao Mei. The BioMediator system as a data integration tool to answer diverse biologic queries. *Stud Health Technol Inform*, 107(Pt 2):768–772, 2004.
- [36] Sarah Cohen-Boulakia, Susan Davidson, Christine Froidevaux, Zoe Lacroix, and Maria-Esther Vidal. Path-based systems to guide scientists in the maze of biological data sources. *J Bioinform Comput Biol*, 4(5):1069–1095, 2006 Oct.
- [37] Aaron Birkland and Golan Yona. Biozon: a system for unification, management and analysis of heterogeneous biological data. *BMC Bioinformatics*, 7:70, 2006.
- [38] Susan B. Davidson, Jonathan Crabtree, Brian P. Brunk, Jonathan Schug, Val Tannen, G. Christian Overton, and Christian J. Stoeckert Jr. K2/Kleisli and GUS: Experiments in integrated access to genomic data sources. *IBM Systems Journal*, 40(2):512–531, 2001.
- [39] I-Min A. Chen and Victor M. Markowitz. An overview of the object-protocol model (opm) and opm data management tools. *Inf. Syst.*, 20(5):393–418, 1995.
- [40] T Etzold, A Ulyanov, and P Argos. Srs: information retrieval system for molecular biology data banks. *Methods Enzymol*, 266:114–128, 1996.
- [41] R Stevens, P Baker, S Bechhofer, G Ng, A Jacoby, N W Paton, C A Goble, and A Brass. Tambis: transparent access to multiple bioinformatics information sources. *Bioinformatics*, 16(2):184–185, 2000 Feb.
- [42] The World Health Report 2008: Now more than ever, 2008.
- [43] David M Morens, Gregory K Folkers, and Anthony S Fauci. Emerging infections: a perpetual challenge. *Lancet Infect Dis*, 8(11):710–719, 2008 Nov.
- [44] G H Cassell. Infectious causes of chronic inflammatory diseases and cancer. *Emerg Infect Dis*, 4(3):475–87, 1998.
- [45] James I Garrels. Yeast genomic databases and the challenge of the post-genomic era. *Funct Integr Genomics*, 2(4-5):212–237, 2002 Sep.
- [46] Stephan Philippi and Jacob Kohler. Addressing the problems with life-science databases for traditional uses and systems biology. *Nat Rev Genet*, 7(6):482–488, 2006 Jun.
- [47] Lawrence Hunter. *Artificial intelligence and molecular biology*. AAAI Press, Menlo Park, Calif., 1993.
- [48] Richard Durbin. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge, UK, 1998.
- [49] Jens Meiler and David Baker. Rapid protein fold determination using unassigned nmr data. *Proc Natl Acad Sci U S A*, 100(26):15404–9, Dec 2003.

- [50] Robert Service. Structural biology. structural genomics, round 2. *Science*, 307(5715):1554–8, Mar 2005.
- [51] John-Marc Chandonia and Steven E Brenner. The impact of structural genomics: expectations and outcomes. *Science*, 311(5759):347–51, Jan 2006.
- [52] S F Altschul, W Gish, W Miller, E W Myers, and D J Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–10, Oct 1990.
- [53] S F Altschul, T L Madden, A A Schäffer, J Zhang, Z Zhang, W Miller, and D J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–402, Sep 1997.
- [54] Joan C. Bartlett and Elaine G. Toms. Developing a protocol for bioinformatics analysis: An integrated information behavior and task analysis approach. *J Am Soc Inform Sci Tech*, 56(5):469–482, Jan 2005.
- [55] K Karplus, C Barrett, and R Hughey. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–56, 1998.
- [56] T Jaakkola, M Diekhans, and D Haussler. A discriminative framework for detecting remote protein homologies. *J Comput Biol*, 7(1-2):95–114, 2000.
- [57] Deepak K Bhalla and David B Warheit. Biological agents with potential for misuse: a historical perspective and defensive measures. *Toxicol Appl Pharmacol*, 199(1):71–84, Aug 2004.
- [58] John M Greene, Frank Collins, Elliot J Lefkowitz, David Roos, Richard H Scheuermann, Bruno Sobral, Rick Stevens, Owen White, and Valentina Di Francesco. National institute of allergy and infectious diseases bioinformatics resource centers: new assets for pathogen informatics. *Infect Immun*, 75(7):3212–9, Jul 2007.
- [59] International travel and health: Situation as on 1 january 2003. Technical report, World Health Organization, January 2003.
- [60] Steven R Gill, Derrick E Fouts, Gordon L Archer, Emmanuel F Mongodin, Robert T Deboy, Jacques Ravel, Ian T Paulsen, James F Kolonay, Lauren Brinkac, Mauren Beanan, Robert J Dodson, Sean C Daugherty, Ramana Madupu, Samuel V Angiuoli, A Scott Durkin, Daniel H Haft, Jessica Vamathevan, Hoda Khouri, Terry Utterback, Chris Lee, George Dimitrov, Lingxia Jiang, Haiying Qin, Jan Weidman, Kevin Tran, Kathy Kang, Ioana R Hance, Karen E Nelson, and Claire M Fraser. Insights on evolution of virulence and resistance from the complete genome analysis of an early methicillin-resistant staphylococcus aureus strain and a biofilm-producing methicillin-resistant staphylococcus epidermidis strain. *J Bacteriol*, 187(7):2426–38, Apr 2005.
- [61] João C Setubal, Marcelo Reis, James Matsunaga, and David A Haake. Lipoprotein computational prediction in spirochaetal genomes. *Microbiology*, 152(Pt 1):113–21, Jan 2006.
- [62] D N Fredericks and D A Relman. Sequence-based identification of microbial pathogens: a reconsideration of koch’s postulates. *Clin Microbiol Rev*, 9(1):18–33, Jan 1996.
- [63] Kelly Paine and Darren R Flower. Bacterial bioinformatics: pathogenesis and the genome. *J Mol Microbiol Biotechnol*, 4(4):357–365, Jul 2002.
- [64] A Casadevall and L A Pirofski. Host-pathogen interactions: redefining the basic concepts of virulence and pathogenicity. *Infect Immun*, 67(8):3703–3713, 1999 Aug.
- [65] Nat F Brown, Mark E Wickham, Brian K Coombes, and B Brett Finlay. Crossing the line: selection and evolution of virulence traits. *PLoS Pathog*, 2(5):e42, May 2006.

- [66] David M Raskin, Rekha Seshadri, Stefan U Pukatzki, and John J Mekalanos. Bacterial genomics and pathogen evolution. *Cell*, 124(4):703–714, 2006 Feb 24.
- [67] L E Alksne and S J Projan. Bacterial virulence as a target for antimicrobial chemotherapy. *Curr Opin Biotechnol*, 11(6):625–636, 2000 Dec.
- [68] Guy R Cornelis. The type iii secretion injectisome. *Nat Rev Microbiol*, 4(11):811–25, Nov 2006.
- [69] B B Finlay and S Falkow. Common themes in microbial pathogenicity revisited. *Microbiol Mol Biol Rev*, 61(2):136–169, Jun 1997.
- [70] Cristina Paiva de Souza. Pathogenicity mechanisms of prokaryotic cells: an evolutionary view. *Braz J Infect Dis*, 7(1):23–31, Feb 2003.
- [71] J.W. Wilson, M.J. Schurr, C.L. LeBlanc, R. Ramamurthy, K.L. Buchanan, and C.A. Nickerson. Mechanisms of bacterial pathogenicity. *Postgrad Med J*, 78:216–224, 2002.
- [72] Steven J Projan. New (and not so new) antibacterial targets - from where and when will the novel drugs come? *Curr Opin Pharmacol*, 2(5):513–22, Oct 2002.
- [73] Hiroaki Suga and Kristina M Smith. Molecular mechanisms of bacterial quorum sensing as a new drug target. *Curr Opin Chem Biol*, 7(5):586–91, Oct 2003.
- [74] J Hacker, G Blum-Oehler, I Mühldorfer, and H Tschäpe. Pathogenicity islands of virulent bacteria: structure, function and impact on microbial evolution. *Mol Microbiol*, 23(6):1089–97, Mar 1997.
- [75] J J Mecsas and E J Strauss. Molecular mechanisms of bacterial virulence: type iii secretion and pathogenicity islands. *Emerg Infect Dis*, 2(4):270–88, 1996.
- [76] E Carniel. The yersinia high-pathogenicity island: an iron-uptake island. *Microbes Infect*, 3(7):561–9, Jun 2001.
- [77] Ryszard Koczura and Adam Kaznowski. The yersinia high-pathogenicity island and iron-uptake systems in clinical isolates of escherichia coli. *J Med Microbiol*, 52(Pt 8):637–42, Aug 2003.
- [78] A Vazquez-Torres, Y Xu, J Jones-Carson, D W Holden, S M Lucia, M C Dinauer, P Mastroeni, and F C Fang. Salmonella pathogenicity island 2-dependent evasion of the phagocyte nadph oxidase. *Science*, 287(5458):1655–8, Mar 2000.
- [79] A M Berry and J C Paton. Additive attenuation of virulence of streptococcus pneumoniae by mutation of the genes encoding pneumolysin and other putative pneumococcal virulence proteins. *Infect Immun*, 68(1):133–40, Jan 2000.
- [80] David J Payne, Michael N Gwynn, David J Holmes, and David L Pompliano. Drugs for bad bugs: confronting the challenges of antibacterial discovery. *Nat Rev Drug Discov*, 6(1):29–40, Jan 2007.
- [81] M Y Galperin and E V Koonin. Searching for drug targets in microbial genomes. *Curr Opin Biotechnol*, 10(6):571–578, 1999 Dec.
- [82] DB Searls. Using bioinformatics in gene and drug discovery. *Drug Discov Today*, 5(4):135–143, 2000 Apr.
- [83] R E Bruccoleri, T J Dougherty, and D B Davison. Concordance analysis of microbial genomes. *Nucleic Acids Res*, 26(19):4482–6, Oct 1998.

- [84] B W Wren. Microbial genome analysis: insights into virulence, host adaptation and evolution. *Nat Rev Genet*, 1(1):30–39, 2000 Oct.
- [85] M G Surette, M B Miller, and B L Bassler. Quorum sensing in *escherichia coli*, *salmonella typhimurium*, and *vibrio harveyi*: a new family of genes responsible for autoinducer production. *Proc Natl Acad Sci U S A*, 96(4):1639–44, Feb 1999.
- [86] S K Mazmanian, G Liu, H Ton-That, and O Schneewind. Staphylococcus aureus sortase, an enzyme that anchors surface proteins to the cell wall. *Science*, 285(5428):760–3, Jul 1999.
- [87] L W Cheng and O Schneewind. Type iii machines of gram-negative bacteria: delivering the goods. *Trends Microbiol*, 8(5):214–20, May 2000.
- [88] R Nordfelth, A M Kauppi, H A Norberg, H Wolf-Watz, and M Elofsson. Small-molecule inhibitors specifically targeting type ”iii” secretion. *Infect Immun*, 73(5):3104–3114, 2005 May.
- [89] Chi-Tai Fang, Yi-Ping Chuang, Chia-Tung Shun, Shan-Chwen Chang, and Jin-Town Wang. A novel virulence gene in *klebsiella pneumoniae* strains causing primary liver abscess and septic metastatic complications. *J Exp Med*, 199(5):697–705, 2004 Mar 1.
- [90] A. Marra. Targeting virulence for antibacterial chemotherapy: identifying and characterizing virulence factors for lead discovery. *Drugs in Research and Design*, 7(1):1–16, 2006.
- [91] Jian Yang, Lihong Chen, Lilian Sun, Jun Yu, and Qi Jin. Vfdb 2008 release: an enhanced web-based resource for comparative pathogenomics. *Nucleic Acids Res*, 36(Database issue):D539–42, Jan 2008.
- [92] Gaurav Sachdeva, Kaushal Kumar, Preti Jain, and Srinivasan Ramachandran. Spaan: a software program for prediction of adhesins and adhesin-like proteins using neural networks. *Bioinformatics*, 21(4):483–491, 2005 Feb 15.
- [93] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: a string kernel for svm protein classification. *Pac Symp Biocomput*, pages 564–75, 2002.
- [94] Sudipto Saha and G P S Raghava. Vicmpred: an svm-based method for the prediction of functional proteins of gram-negative bacteria using amino acid patterns and composition. *Genomics Proteomics Bioinformatics*, 4(1):42–7, Feb 2006.
- [95] Aarti Garg and Dinesh Gupta. Virulentpred: a svm based prediction method for virulent proteins in bacterial pathogens. *BMC Bioinformatics*, 9:62, 2008.
- [96] P Bork and E V Koonin. Predicting functions from protein sequences—where are the bottlenecks? *Nat Genet*, 18(4):313–318, 1998 Apr.
- [97] Zoé Lacroix and Terence Critchlow. *Bioinformatics: managing scientific data*. Morgan Kaufmann Publishers, San Francisco, CA, 2003.
- [98] William A. Baumgartner Jr., K. Bretonnel Cohen, Lynne M. Fox, George Acquaaah-Mensah, and Lawrence Hunter. Manual curation is not sufficient for annotation of genomic databases. In *ISMB/ECCB (Supplement of Bioinformatics)*, pages 41–48, 2007.
- [99] Takeya Kasukawa, Masaaki Furuno, Itoshi Nikaido, Hidemasa Bono, David A Hume, Carol Bult, David P Hill, Richard Baldarelli, Julian Gough, Alexander Kanapin, Hideo Matsuda, Lynn M Schriml, Yoshihide Hayashizaki, Yasushi Okazaki, and John Quackenbush. Development and evaluation of an automated annotation pipeline and cdna annotation system. *Genome Res*, 13(6B):1542–51, Jun 2003.

- [100] Simon C Potter, Laura Clarke, Val Curwen, Stephen Keenan, Emmanuel Mongin, Stephen M J Searle, Arne Stabenau, Roy Storey, and Michele Clamp. The ensembl analysis pipeline. *Genome Res*, 14(5):934–41, May 2004.
- [101] Philippe Gouret, Verane Vitiello, Nathalie Balandraud, Andre Gilles, Pierre Pontarotti, and Etienne G J Danchin. Figenix: intelligent automation of genomic annotation: expertise integration in a new software platform. *BMC Bioinformatics*, 6:198, 2005.
- [102] M Ashburner, C A Ball, J A Blake, D Botstein, H Butler, J M Cherry, A P Davis, K Dolinski, S S Dwight, J T Eppig, M A Harris, D P Hill, L Issel-Tarver, A Kasarskis, S Lewis, J C Matese, J E Richardson, M Ringwald, G M Rubin, and G Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–9, May 2000.
- [103] Steffen Hennig, Detlef Groth, and Hans Lehrach. Automated gene ontology annotation for anonymous sequence data. *Nucleic Acids Res*, 31(13):3712–5, Jul 2003.
- [104] Brenton Louie, Landon Detwiler, Nilesh N. Dalvi, Ron Shaker, Peter Tarczy-Hornoch, and Dan Suci. Incorporating uncertainty metrics into a general-purpose data integration system. In *SSDBM* [263], page 19.
- [105] Peter Karp. What we do not know about sequence analysis and sequence databases. *Bioinformatics*, 14(9):753–754, 1998.
- [106] S E Brenner. Errors in genome annotation. *Trends Genet*, 15(4):132–133, 1999 Apr.
- [107] Walter R Gilks, Benjamin Audit, Daniela De Angelis, Sophia Tsoka, and Christos A Ouzounis. Modeling the percolation of annotation errors in a database of protein sequences. *Bioinformatics*, 18(12):1641–9, Dec 2002.
- [108] Heiko Müller. Semantic data cleansing in genome databases. In Scholl and Grust [217].
- [109] Xiaotu Ma, Hyunju Lee, Li Wang, and Fengzhu Sun. Cgi: a new approach for prioritizing genes by combining gene expression and protein-protein interaction data. *Bioinformatics*, 23(2):215–21, Jan 2007.
- [110] Brenton Louie, Peter Mork, Fernando Martin-Sanchez, Alon Halevy, and Peter Tarczy-Hornoch. Data integration and genomic medicine. *J Biomed Inform*, 40(1):5–16, Feb 2007.
- [111] Limsoon Wong. Technologies for integrating biological data. *Brief Bioinform*, 3(4):389–404, Dec 2002.
- [112] Oliver D King, Rebecca E Foulger, Selina S Dwight, James V White, and Frederick P Roth. Predicting gene function from patterns of annotation. *Genome Res*, 13(5):896–904, May 2003.
- [113] Oliver D King, Jeffrey C Lee, Aimée M Dudley, Daniel M Janse, George M Church, and Frederick P Roth. Predicting phenotype from patterns of annotation. *Bioinformatics*, 19 Suppl 1:i183–9, 2003.
- [114] Olga G Troyanskaya, Kara Dolinski, Art B Owen, Russ B Altman, and David Botstein. A bayesian framework for combining heterogeneous data sources for gene function prediction (in *saccharomyces cerevisiae*). *Proc Natl Acad Sci U S A*, 100(14):8348–53, Jul 2003.
- [115] Minghua Deng, Ting Chen, and Fengzhu Sun. An integrated probabilistic model for functional prediction of proteins. *J Comput Biol*, 11(2-3):463–75, 2004.
- [116] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, pages 144–152, 1992.
- [117] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

- [118] Trevor Hastie, Robert Tibshirani, and J. H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, New York, 2001.
- [119] M P Brown, W N Grundy, D Lin, N Cristianini, C W Sugnet, T S Furey, M Ares, Jr, and D Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc Natl Acad Sci U S A*, 97(1):262–7, Jan 2000.
- [120] S Hua and Z Sun. A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. *J Mol Biol*, 308(2):397–407, Apr 2001.
- [121] Christina S Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–76, Mar 2004.
- [122] Paul Pavlidis, Jinsong Cai, Jason Weston, and William N. Grundy. Gene functional classification from heterogeneous data. In *Proceedings of the Fifth International Conference on Computational Molecular Biology*, number 5 in RECOMB. ISCB, April 2001.
- [123] Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert. *Kernel methods in computational biology*. MIT Press, Cambridge, Mass., 2004.
- [124] Anneleen Daemen, Olivier Gevaert, and Bart De Moor. Integration of clinical and microarray data with kernel methods. *Conf Proc IEEE Eng Med Biol Soc*, 2007:5411–5, 2007.
- [125] Gert R G Lanckriet, Tijl De Bie, Nello Cristianini, Michael I Jordan, and William Stafford Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004 Nov 1.
- [126] Kosuke Fujishima, Mizuki Komasa, Sayaka Kitamura, Haruo Suzuki, Masaru Tomita, and Akio Kanai. Proteome-wide prediction of novel dna/rna-binding proteins using amino acid composition and periodicity in the hyperthermophilic archaeon *pyrococcus furiosus*. *DNA Res*, 14(3):91–102, Jun 2007.
- [127] Anneleen Daemen, Olivier Gevaert, Fabian Ojeda, Annelies Debucquoy, Johan Ak Suykens, Christine Sempoux, Jean-Pascal Machiels, Karin Haustermans, and Bart De Moor. A kernel-based integration of genome-wide data for clinical decision support. *Genome Med*, 1(4):39, 2009.
- [128] Guillaume Obozinski, Gert Lanckriet, Charles Grant, Michael I Jordan, and William Stafford Noble. Consistent probabilistic outputs for protein function prediction. *Genome Biol*, 9 Suppl 1:S6, 2008.
- [129] Lincoln D Stein. Integrating biological databases. *Nat Rev Genet*, 4(5):337–45, May 2003.
- [130] Maurizio Lenzerini. Data integration: A theoretical perspective. In Popa [248], pages 233–246.
- [131] Kelan Wang. Validating generalizability and extensibility of the BioMediator system for data integration in life science. Master’s thesis, University of Washington, 2005.
- [132] P Mork, A Halevy, and P Tarczy-Hornoch. A model for data integration systems of biomedical data applied to online genetic databases. *Proc AMIA Symp*, pages 473–477, 2001.
- [133] Zoé Lacroix, Hyma Murthy, Felix Naumann, and Louiqa Raschid. Links and paths through life sciences data sources. In Rahm [245], pages 203–211.
- [134] Zoe Lacroix, Louiqa Raschid, and Maria-Esther Vidal. Efficient techniques to explore and rank paths in life science data sources. In Rahm [245], pages 187–202.

- [135] Peter Mork. *Peer Architectures for Knowledge Sharing*. PhD thesis, University of Washington, 2005.
- [136] H Mei, P Tarczy-Hornoch, P Mork, A J Rossini, R Shaker, and L Donelson. Expression array annotation using the biomediator biological data integration system and the bioconductor analytic platform. *AMIA Annu Symp Proc*, pages 445–449, 2003.
- [137] K Wang, P Tarczy-Hornoch, R Shaker, P Mork, and J F Brinkley. Biomediator data integration: beyond genomics to neuroscience data. *AMIA Annu Symp Proc*, pages 779–783, 2005.
- [138] Zoé Lacroix, Kaushal Parekh, Maria-Esther Vidal, Marelis Cardenas, and Natalia Marquez. Bionavigation: Selecting optimum paths through biological resources to evaluate ontological navigational queries. In Ludäscher and Raschid [252], pages 275–283.
- [139] Jeffrey D. Ullman. Information integration using logical views. In Afrati and Kolaitis [246], pages 19–40.
- [140] Richard Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *PODS* [247], pages 51–61.
- [141] Guido Van Rossum. Python programming language.
- [142] Eithon Cadag and Peter Tarczy-Hornoch. Supporting retrieval of diverse biomedical data using evidence-aware queries. *AMIA Annu Symp Proc*, (To appear), November 2009.
- [143] Seth Carbon, Amelia Ireland, Christopher J Mungall, ShengQiang Shu, Brad Marshall, and Suzanna Lewis. Amigo: online access to ontology and annotation data. *Bioinformatics*, 25(2):288–289, 2009 Jan 15.
- [144] Peter D Karp, Christos A Ouzounis, Caroline Moore-Kochlacs, Leon Goldovsky, Pallavi Kaipa, Dag Ahren, Sophia Tsoka, Nikos Darzentas, Victor Kunin, and Nuria Lopez-Bigas. Expansion of the biocyc collection of pathway/genome databases to 160 genomes. *Nucleic Acids Res*, 33(19):6083–6089, 2005.
- [145] Aron Marchler-Bauer, John B Anderson, Farideh Chitsaz, Myra K Derbyshire, Carol DeWeese-Scott, Jessica H Fong, Lewis Y Geer, Renata C Geer, Noreen R Gonzales, Marc Gwadz, Siqian He, David I Hurwitz, John D Jackson, Zhaoxi Ke, Christopher J Lanczycki, Cynthia A Liebert, Chunlei Liu, Fu Lu, Shennan Lu, Gabriele H Marchler, Mikhail Mullokandov, James S Song, Asba Tasneem, Narmada Thanki, Roxanne A Yamashita, Dachuan Zhang, Naigong Zhang, and Stephen H Bryant. Cdd: specific functional annotation with the conserved domain database. *Nucleic Acids Res*, 37(Database issue):D205–10, 2009 Jan.
- [146] Sarah Hunter, Rolf Apweiler, Teresa K Attwood, Amos Bairoch, Alex Bateman, David Binns, Peer Bork, Ujjwal Das, Louise Daugherty, Lauranne Duquenne, Robert D Finn, Julian Gough, Daniel Haft, Nicolas Hulo, Daniel Kahn, Elizabeth Kelly, Aurelie Laugraud, Ivica Letunic, David Lonsdale, Rodrigo Lopez, Martin Madera, John Maslen, Craig McAnulla, Jennifer McDowall, Jaina Mistry, Alex Mitchell, Nicola Mulder, Darren Natale, Christine Orengo, Antony F Quinn, Jeremy D Selengut, Christian J A Sigrist, Manjula Thimma, Paul D Thomas, Franck Valentin, Derek Wilson, Cathy H Wu, and Corin Yeats. Interpro: the integrative protein signature database. *Nucleic Acids Res*, 37(Database issue):D211–5, 2009 Jan.
- [147] Kiyoko F Aoki and Minoru Kanehisa. Using the kegg database resource. *Curr Protoc Bioinformatics*, Chapter 1:Unit 1.12, 2005 Oct.
- [148] J.L. Sussman, D. Lin, J. Jiang, N.O. Manning, J. Prilusky, O. Ritter, and E.E. Abola. Protein Data Bank (PDB): Database of three-dimensional structural information of biological macromolecules. *Acta Crystallographica*, 54(1):1078–1084, Nov 1998.

- [149] Daniel H Haft, Jeremy D Selengut, and Owen White. The tigrfams database of protein families. *Nucleic Acids Res*, 31(1):371–373, 2003 Jan 1.
- [150] Donna Maglott, Jim Ostell, Kim D Pruitt, and Tatiana Tatusova. Entrez gene: gene-centered information at ncbi. *Nucleic Acids Res*, 35(Database issue):D26–31, 2007 Jan.
- [151] Dennis A Benson, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. Genbank. *Nucleic Acids Res*, 37(Database issue):D26–31, 2009 Jan.
- [152] Emmanuel Boutet, Damien Lieberherr, Michael Tognolli, Michel Schneider, and Amos Bairoch. Uniprotkb/swiss-prot. *Methods Mol Biol*, 406:89–112, 2007.
- [153] P Mork, R Shaker, A Halevy, and P Tarczy-Hornoch. Pql: a declarative query language over dynamic biological schemata. *Proc AMIA Symp*, pages 533–537, 2002.
- [154] Peter Tarczy-Hornoch, Peter Mork, and Ron Shaker. BioMediator data integration system.
- [155] Larry Wall. Perl programming language.
- [156] James Gosling. Java programming language.
- [157] Todd J. Green, Gregory Karvounarakis, Nicholas E. Taylor, Olivier Biton, Zachary G. Ives, and Val Tannen. Orchestra: facilitating collaborative data sharing. In Chan et al. [205], pages 1131–1133.
- [158] Eithon Cadag, Peter Tarczy-Hornoch, and Peter J. Myler. On the reachability of trustworthy information from integrated exploratory biological queries. In Paton et al. [219], pages 55–70.
- [159] R Albert, H Jeong, and AL Barabasi. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000 Jul 27.
- [160] L A Amaral, A Scala, M Barthelemy, and H E Stanley. Classes of small-world networks. *Proc Natl Acad Sci U S A*, 97(21):11149–11152, 2000 Oct 10.
- [161] David B Searls. Data integration—connecting the dots. *Nat Biotechnol*, 21(8):844–845, 2003 Aug.
- [162] Krishna Bharat and Monika Rauch Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *SIGIR* [208], pages 104–111.
- [163] Koji Tsuda and William Stafford Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20 Suppl 1:i326–33, 2004 Aug 4.
- [164] Jason Weston, Andre Elisseeff, Dengyong Zhou, Christina S Leslie, and William Stafford Noble. Protein ranking: from local to global structure in the protein similarity network. *Proc Natl Acad Sci U S A*, 101(17):6559–6563, 2004 Apr 27.
- [165] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.
- [166] Olivier Bodenreider. Gennav: Visualizing gene ontology as a graph. *Proc AMIA Ann Symp*, 2002.
- [167] Tom Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. Technical report, Hewlett Packard Laboratories, March 2004.
- [168] Foster J. Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.
- [169] AE Hoerl. Application of ridge analysis to regression problems. *Chem. Eng. Progr.*, 1962.

- [170] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inform. Th.*, 13(1):21–27, Jan 1967.
- [171] Pseudomonas Genome Database, Jan 2009.
- [172] Comprehensive Microbial Resource (JVC1), Jan 2009.
- [173] B B Finlay and S Falkow. Common themes in microbial pathogenicity revisited. *Microbiol Mol Biol Rev*, 61(2):136–169, 1997 Jun.
- [174] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–9, Jul 2006.
- [175] Tonia Korves and Marc E Colosimo. Controlled vocabularies for microbial virulence factors. *Trends Microbiol*, 17(7):279–85, Jul 2009.
- [176] D A Lindberg, B L Humphreys, and A T McCray. The unified medical language system. *Methods Inf Med*, 32(4):281–91, Aug 1993.
- [177] Cornelius Rosse and José L V Mejino, Jr. A reference ontology for biomedical informatics: the foundational model of anatomy. *J Biomed Inform*, 36(6):478–500, Dec 2003.
- [178] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In Apers et al. [207], pages 49–58.
- [179] Songmao Zhang and Olivier Bodenreider. Experience in aligning anatomical ontologies. *Int J Semant Web Inf Syst*, 3(2):1–26, 2007.
- [180] Li Liao and William Stafford Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *RECOMB*, pages 225–232, 2002.
- [181] Asa Ben-Hur and Douglas Brutlag. Remote homology detection: a motif based approach. *Bioinformatics*, 19 Suppl 1:i26–33, 2003.
- [182] Yi-Wei Chen and Chih-Jen Lin. Combining svms with various feature selection strategies. In *Feature Extraction: Foundations and Applications*, volume 207 of *Studies in Fuzziness and Soft Computing*. Springer, November 2006.
- [183] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*.
- [184] P Bork. Powers and pitfalls in sequence analysis: the 70% hurdle. *Genome Res*, 10(4):398–400, 2000 Apr.
- [185] M Gribskov and N L Robinson. Use of receiver operating characteristic (roc) analysis to evaluate sequence matching. *Comput Chem*, 20(1):25–33, Mar 1996.
- [186] Volker Roth and Bernd Fischer. Improved functional prediction of proteins by learning kernel combinations in multilabel settings. *BMC Bioinformatics*, 8 Suppl 2:S12, 2007.
- [187] Terry Hsin-Yi Shen. *Determining the Feasibility and Value of Federated Data Integration with Combinations of Logical and Probabilistic Inference for SNP Annotation*. PhD thesis, University of Washington, 2009.
- [188] N J White. Melioidosis. *Lancet*, 361(9370):1715–1722, 2003 May 17.
- [189] Apichai Tuanyok, H Stanley Kim, William C Nierman, Yan Yu, John Dunbar, Richard A Moore, Patricia Baker, Marina Tom, Jessmi M L Ling, and Donald E Woods. Genome-wide expression analysis of iron regulation in burkholderia pseudomallei and burkholderia mallei using dna microarrays. *FEMS Microbiol Lett*, 252(2):327–35, Nov 2005.

- [190] Matthew D Dyer, T M Murali, and Bruno W Sobral. The landscape of human proteins interacting with viruses and other pathogens. *PLoS Pathog*, 4(2):e32, Feb 2008.
- [191] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Dietterich et al. [193], pages 841–848.
- [192] Alexander R. Statnikov, Lily Wang, and Constantin F. Aliferis. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC Bioinformatics*, 9, 2008.
- [193] Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors. *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. MIT Press, 2001.
- [194] Protégé.
- [195] Baolin Wu, Tom Abbott, David Fishman, Walter McMurray, Gil Mor, Kathryn Stone, David Ward, Kenneth Williams, and Hongyu Zhao. Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. *Bioinformatics*, 19(13):1636–1643, 2003 Sep 1.
- [196] James A Casbon and Mansoor A S Saqi. On single and multiple models of protein families for the detection of remote sequence relationships. *BMC Bioinformatics*, 7:48, 2006.
- [197] A Biegert and J Soding. Sequence context-specific profiles for homology searching. *Proc Natl Acad Sci U S A*, 106(10):3770–3775, 2009 Mar 10.
- [198] Philip E. S Palmer, Maurice M Reeder, and Maurice M Reeder. *Imaging of tropical diseases: with epidemiological, pathological, and clinical correlation*. Springer, Heidelberg, Germany, 2nd ed., rev edition, 2001.
- [199] Cdc: Melioidosis general information.
- [200] Allen C Cheng and Bart J Currie. Melioidosis: epidemiology, pathophysiology, and management. *Clin Microbiol Rev*, 18(2):383–416, 2005 Apr.
- [201] Niaid category a, b, and c priority pathogens, Sept 2009.
- [202] Jung-Hsien Chiang and Hsu-Chun Yu. Meke: discovering the functions of gene products from biomedical literature via sentence alignment. *Bioinformatics*, 19(11):1417–1422, 2003 Jul 22.
- [203] Evelyn Camon, Michele Magrane, Daniel Barrell, David Binns, Wolfgang Fleischmann, Paul Kersey, Nicola Mulder, Tom Oinn, John Maslen, Anthony Cox, and Rolf Apweiler. The gene ontology annotation (goa) project: implementation of go in swiss-prot, trembl, and interpro. *Genome Res*, 13(4):662–672, 2003 Apr.
- [204] J A Mitchell, A T McCray, and O Bodenreider. From phenotype to genotype: issues in navigating the available information resources. *Methods Inf Med*, 42(5):557–563, 2003.
- [205] Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou, editors. *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*. ACM, 2007.
- [206] Brigitte Boeckmann, Amos Bairoch, Rolf Apweiler, Marie-Claude Blatter, Anne Estreicher, Elisabeth Gasteiger, Maria J Martin, Karine Michoud, Claire O’Donovan, Isabelle Phan, Sandrine Pilbout, and Michel Schneider. The swiss-prot protein knowledgebase and its supplement trembl in 2003. *Nucleic Acids Res*, 31(1):365–70, Jan 2003.

- [207] Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard T. Snodgrass, editors. *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*. Morgan Kaufmann, 2001.
- [208] *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*. ACM, 1998.
- [209] Alfonso Valencia. Automatic annotation of protein function. *Curr Opin Struct Biol*, 15(3):267–74, Jun 2005.
- [210] Luis Marenco, Tzuu-Yi Wang, Gordon Shepherd, Perry L Miller, and Prakash Nadkarni. Qis: A framework for biomedical database federation. *J Am Med Inform Assoc*, 11(6):523–534, 2004 Nov-Dec.
- [211] Nathan Bales, James Brinkley, E. Sally Lee, Shobhit Mathur, Christopher Re, and Dan Suci. A framework for xml-based integration of data, visualization and analysis in a biomedical domain. In Bressan et al. [212], pages 207–221.
- [212] Stéphane Bressan, Stefano Ceri, Ela Hunt, Zachary G. Ives, Zohra Bellahsene, Michael Rys, and Rainer Unland, editors. *Database and XML Technologies, Third International XML Database Symposium, XSym 2005, Trondheim, Norway, August 28-29, 2005, Proceedings*, volume 3671 of *Lecture Notes in Computer Science*. Springer, 2005.
- [213] Richard M. Karp and Michael Luby. Monte-carlo algorithms for enumeration and reliability problems. In *FOCS* [214], pages 56–64.
- [214] *24th Annual Symposium on Foundations of Computer Science, 7-9 November 1983, Tucson, Arizona, USA*. IEEE, 1983.
- [215] Thomas Hernandez and Subbarao Kambhampati. Integration of biological sources: Current systems and challenges ahead. *SIGMOD Record*, 33(3):51–60, 2004.
- [216] *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*. IEEE, 2009.
- [217] Marc H. Scholl and Torsten Grust, editors. *Proceedings of the VLDB 2003 PhD Workshop. Co-located with the 29th International Conference on Very Large Data Bases (VLDB 2003). Berlin, September 12-13, 2003*, volume 76 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.
- [218] D Devos and A Valencia. Intrinsic errors in genome annotation. *Trends Genet*, 17(8):429–31, Aug 2001.
- [219] Norman W. Paton, Paolo Missier, and Cornelia Hedeler, editors. *Data Integration in the Life Sciences, 6th International Workshop, DILS 2009, Manchester, UK, July 20-22, 2009. Proceedings*, volume 5647 of *Lecture Notes in Computer Science*. Springer, 2009.
- [220] Sébastien Rey, Jennifer L Gardy, and Fiona S L Brinkman. Assessing the precision of high-throughput computational and laboratory approaches for the genome-wide identification of protein subcellular localization in bacteria. *BMC Genomics*, 6:162, 2005.
- [221] U Dobrindt and J Hacker. Whole genome plasticity in pathogenic bacteria. *Curr Opin Microbiol*, 4(5):550–7, Oct 2001.
- [222] Eugene Rosenberg. The diversity of bacterial pathogenicity mechanisms. *Genome Biol*, 6(5):320, 2005.

- [223] Quan-Yuan He, Quan-Ze He, Xing-Can Deng, Lei Yao, Er Meng, Zhong-Hua Liu, and Song-Ping Liang. Atldb: a uni-database platform for animal toxins. *Nucleic Acids Res*, 36(Database issue):D293–7, Jan 2008.
- [224] Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21 Suppl 1:i38–46, Jun 2005.
- [225] Carol L Ecale Zhou, Marisa W Lam, Jason R Smith, Adam T Zemla, Matthew D Dyer, Thomas A Kuczmarski, Elizabeth A Vitalis, and Thomas R Slezak. Manndb - a microbial database of automated protein sequence analyses and evidence integration for protein characterization. *BMC Bioinformatics*, 7:459, 2006.
- [226] Rainer Winnenburger, Martin Urban, Andrew Beacham, Thomas K Baldwin, Sabrina Holland, Magdalen Lindeberg, Hilde Hansen, Christopher Rawlings, Kim E Hammond-Kosack, and Jacob Köhler. Phi-base update: additions to the pathogen host interaction database. *Nucleic Acids Res*, 36(Database issue):D572–6, Jan 2008.
- [227] Sudipto Saha and Gajendra P S Raghava. Predicting virulence factors of immunological interest. *Methods Mol Biol*, 409:407–15, 2007.
- [228] Yang Zhang. Progress and challenges in protein structure prediction. *Curr Opin Struct Biol*, 18(3):342–8, Jun 2008.
- [229] C A Wilson, J Kreychman, and M Gerstein. Assessing annotation transfer for genomics: quantifying the relations between protein sequence, structure and function through traditional and probabilistic scores. *J Mol Biol*, 297(1):233–49, Mar 2000.
- [230] D Eisenberg, E M Marcotte, I Xenarios, and T O Yeates. Protein function in the post-genomic era. *Nature*, 405(6788):823–6, Jun 2000.
- [231] Mickael Desvaux, Nicholas J Parham, Anthony Scott-Tucker, and Ian R Henderson. The general secretory pathway: a general misnomer? *Trends Microbiol*, 12(7):306–309, Jul 2004.
- [232] Raphael Hoffmann. Model Selection for Support Vector Machines. Master’s thesis, Universität Passau, Passau, Germany, August 2005.
- [233] Iain Melvin, Jason Weston, Christina S Leslie, and William S Noble. Combining classifiers for improved classification of proteins from sequence or structure. *BMC Bioinformatics*, 9:389, 2008.
- [234] Doina Caragea, Jun Zhang 0002, Jyotishman Pathak, and Vasant Honavar. Learning classifiers from distributed, ontology-extended data sources. In Tjoa and Trujillo [235], pages 363–373.
- [235] A. Min Tjoa and Juan Trujillo, editors. *Data Warehousing and Knowledge Discovery, 8th International Conference, DaWaK 2006, Krakow, Poland, September 4-8, 2006, Proceedings*, volume 4081 of *Lecture Notes in Computer Science*. Springer, 2006.
- [236] Doina Caragea, Jie Bao, Jyotishman Pathak, Adrian Silvescu, Carson M. Andorf, Drena Dobbs, and Vasant Honavar. Information integration from semantically heterogeneous biological data sources. In *DEXA Workshops* [237], pages 580–584.
- [237] *16th International Workshop on Database and Expert Systems Applications (DEXA 2005), 22-26 August 2005, Copenhagen, Denmark*. IEEE Computer Society, 2005.
- [238] Anton J Enright, Victor Kunin, and Christos A Ouzounis. Protein families and tribes in genome sequence space. *Nucleic Acids Res*, 31(15):4632–4638, 2003 Aug 1.
- [239] T M Murali, Chang-Jiun Wu, and Simon Kasif. The art of gene function prediction. *Nat Biotechnol*, 24(12):1474–1475, 2006 Dec.

- [240] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy. Gene functional classification from heterogeneous data. In *RECOMB*, pages 249–255, 2001.
- [241] M.E.J. Newman. The structure and function of complex networks. *SIAM Rev*, 45(2):167–256, 2003.
- [242] Gerard Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.
- [243] Jason Weston, Rui Kuang, Christina Leslie, and William Stafford Noble. Protein ranking by semi-supervised network propagation. *BMC Bioinformatics*, 7 Suppl 1:S10, 2006.
- [244] Eithon Cadag, Peter Tarczy-Hornoch, and Peter J Myler. Learning pathogenic proteins across fractured and heterogeneous data. *AMIA Annu Symp Proc*, page 889, 2008.
- [245] Erhard Rahm, editor. *Data Integration in the Life Sciences, First International Workshop, DILS 2004, Leipzig, Germany, March 25-26, 2004, Proceedings*, volume 2994 of *Lecture Notes in Computer Science*. Springer, 2004.
- [246] Foto N. Afrati and Phokion G. Kolaitis, editors. *Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings*, volume 1186 of *Lecture Notes in Computer Science*. Springer, 1997.
- [247] *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona*. ACM Press, 1997.
- [248] Lucian Popa, editor. *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*. ACM, 2002.
- [249] Zoe Lacroix, Tiffany Morris, Kaushal Parekh, Louiqa Raschid, and Maria-Esther Vidal. Exploiting multiple paths to express scientific queries. In *SSDBM* [250], pages 357–.
- [250] *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004), 21-23 June 2004, Santorini Island, Greece*. IEEE Computer Society, 2004.
- [251] Sarah Cohen Boulakia, Susan B. Davidson, and Christine Froidevaux. A user-centric framework for accessing biological sources and tools. In Ludäscher and Raschid [252], pages 3–18.
- [252] Bertram Ludäscher and Louiqa Raschid, editors. *Data Integration in the Life Sciences, Second International Workshop, DILS 2005, San Diego, CA, USA, July 20-22, 2005, Proceedings*, volume 3615 of *Lecture Notes in Computer Science*. Springer, 2005.
- [253] Christos A Ouzounis and Peter D Karp. The past, present and future of genome-wide re-annotation. *Genome Biol*, 3(2):COMMENT2001, 2002.
- [254] C Sansom. Database searching with dna and protein sequences: an introduction. *Brief Bioinform*, 1(1):22–32, 2000 Feb.
- [255] D. James Harris. Can you bank on genbank? *Trends in Eco and Evo*, 18(7):317–319, July 2003.
- [256] Laura S Burrack and Darren E Higgins. Genomic approaches to understanding bacterial virulence. *Curr Opin Microbiol*, 10(1):4–9, 2007 Feb.
- [257] Paul Stothard and David S Wishart. Automated bacterial genome analysis and annotation. *Curr Opin Microbiol*, 9(5):505–510, 2006 Oct.

- [258] H B Tang, E DiMango, R Bryan, M Gambello, B H Iglewski, J B Goldberg, and A Prince. Contribution of specific pseudomonas aeruginosa virulence factors to pathogenesis of pneumonia in a neonatal mouse model of infection. *Infect Immun*, 64(1):37–43, 1996 Jan.
- [259] M H Serres, S Gopal, L A Nahum, P Liang, T Gaasterland, and M Riley. A functional update of the escherichia coli k-12 genome. *Genome Biol*, 2(9):RESEARCH0035, 2001.
- [260] G H Thomas. Completing the e. coli proteome: a database of gene products characterised since the completion of the genome sequence. *Bioinformatics*, 15(10):860–861, 1999 Oct.
- [261] The world health report 2007 - a safer future: global public health security in the 21st century, 2008.
- [262] Colin D Mathers and Dejan Loncar. Projections of global mortality and burden of disease from 2002 to 2030. *PLoS Med*, 3(11):e442, 2006 Nov.
- [263] *19th International Conference on Scientific and Statistical Database Management, SSDBM 2007, 9-11 July 2007, Banff, Canada, Proceedings*. IEEE Computer Society, 2007.
- [264] Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Tiffany Murray, and Teri E. Klein, editors. *Biocomputing 2007, Proceedings of the Pacific Symposium, Maui, Hawaii, USA, 3-7 January 2007*. World Scientific, 2007.
- [265] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Stafford Noble. Learning gene functional classifications from multiple data types. *J Comput Biol*, 9(2):401–411, 2002.
- [266] David M Morens, Gregory K Folkers, and Anthony S Fauci. The challenge of emerging and re-emerging infectious diseases. *Nature*, 430(6996):242–249, 2004 Jul 8.
- [267] Aaron Clauset, Cosma R. Shalizi, and M.E.J. Newman. Power-law distributions in empirical data. arXiv:0706.1062v1, 2007.
- [268] Vanessa DCosta, Katherine M. McGrann, Donald W. Hughes, and Gerard D. Wright. Sampling the antibiotic resistome. *Science*, 311(5759):374–377, 2006.

Vita

Eithon Cadag was born in Seattle, Washington, and completed his Bachelor of Science in Informatics and Bachelor of Arts in Management Information Systems at the University of Washington. He stayed on at the university, further completing a Masters of Science in Biomedical and Health Informatics while working with Drs. Peter Myler and Peter Tarczy-Hornoch. His research activities have included ubiquitous computing for both scientific and contextual localization purposes, gene identification and annotation, presenting and visualizing clinical drug compliance data and natural language processing over medical notes. Eithon's current research interests lay at the intersections between biomedicine and large-scale data management, integration and knowledge discovery.