

**Understanding the Hormonal Regulation of  
Mouse Lactogenesis by  
Transcriptomics and Literature Analysis**

Maurice Han Tong LING, BSc(Hons)

Submitted in total fulfilment of the requirements  
of the degree of Doctor of Philosophy

January 2010

Department of Zoology  
Faculty of Science  
The University of Melbourne  
Australia



## Abstract

The mammary explant culture model has been a major experimental tool for studying hormonal requirements for milk protein gene expression as markers of secretory differentiation. Experiments with mammary explants from pregnant animals from many species have established that insulin, prolactin, and glucocorticoid are the minimal set of hormones required for the induction of maximal milk protein gene expression. However, the extent to which mammary explants mimic the response of the mammary gland in vivo is not clear. Recent studies have used microarray technology to study the transcriptome of mouse lactation cycle. It was demonstrated that the each phase of mouse lactation has a distinct transcriptional profile but making sense of microarray results requires analysis of large amounts of biological information which is increasingly difficult to access as the amount of literature increases.

The first objective is to examine the possibility of combining literature and genomic analysis to elucidate potentially novel hypotheses for further research into lactation biology. The second objective is to evaluate the strengths and limitations of the murine mammary explant culture for the study and understanding of murine lactogenesis. The underlying question to this objective is whether the mouse mammary explant culture is a good model or representation to study mouse lactogenesis.

The exponential increase in publication rate of new articles is limiting access of researchers to relevant literature. This has prompted the use of text mining tools to extract key biological information. Previous studies have reported extensive modification of existing generic text processors to process biological text. However, this requirement for modification had not been examined. We have constructed Muscorian, using MontyLingua, a generic text processor. It uses a two-layered generalization-specialization paradigm previously proposed where text was generically processed to a suitable intermediate format before domain-specific data extraction techniques are applied at the specialization layer. Evaluation using a corpus and experts indicated 86-90% precision and approximately 30% recall in extracting protein-protein interactions, which was comparable to previous studies using either specialized biological text processing tools or modified existing tools. This study also demonstrated the flexibility

of the two-layered generalization-specialization paradigm by using the same generalization layer for two specialized information extraction tasks.

The performance of Muscorian was unexpected since potential errors from a series of text analysis processes is likely to adversely affect the outcome of the entire process. Most biomedical entity relationship extraction tools have used biomedical-specific parts-of-speech (POS) tagger as errors in POS tagging and are likely to affect subsequent semantic analysis of the text, such as shallow parsing. A comparative study between MontyTagger, a generic POS tagger, and MedPost, a tagger trained in biomedical text, was carried out. Our results demonstrated that MontyTagger, Muscorian's POS tagger, has a POS tagging accuracy of 83.1% when tested on biomedical text. Replacing MontyTagger with MedPost did not result in a significant improvement in entity relationship extraction from text; precision of 55.6% from MontyTagger versus 56.8% from MedPost on directional relationships and 86.1% from MontyTagger compared to 81.8% from MedPost on un-directional relationships. This is unexpected as the potential for poor POS tagging by MontyTagger is likely to affect the outcome of the information extraction. An analysis of POS tagging errors demonstrated that 78.5% of tagging errors are being compensated by shallow parsing. Thus, despite 83.1% tagging accuracy, MontyTagger has a functional tagging accuracy of 94.6%. This suggests that POS tagging error does not adversely affect the information extraction task if the errors were resolved in shallow parsing through alternative POS tag use.

Microarrays had been used to examine the transcriptome of mouse lactation and a simple method for microarray analysis is correlation studies where functionally related genes exhibit similar expression profiles. However, there has been no study to date using text mining to sieve microarray analysis to generate new hypotheses for further research in the field of lactational biology. Our results demonstrated that a previously reported protein name co-occurrence method (5-mention PubGene) which was not based on a hypothesis testing framework, is generally more stringent than the 99<sup>th</sup> percentile of Poisson distribution-based method of calculating co-occurrence. It agrees with previous methods using natural language processing to extract protein-protein interaction from text as more than 96% of the interactions found by natural language processing methods coincide with the results from 5-mention PubGene method. However, less than 2% of the gene co-expressions analyzed by microarray were found

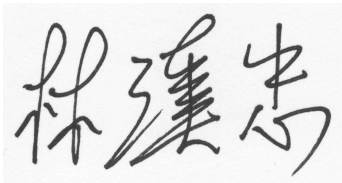
from direct co-occurrence or interaction information extraction from the literature. At the same time, combining microarray and literature analyses, we derive a novel set of 7 potential functional protein-protein interactions that had not been previously described in the literature. We conclude that the 5-mention PubGene method is more stringent than the 99<sup>th</sup> percentile of Poisson distribution method for extracting protein-protein interactions by co-occurrence of entity names and literature analysis may be a potential filter for microarray analysis to isolate potentially novel hypotheses for further research.

The availability of transcriptomics data from time-course experiments on mouse mammary glands examined during the lactation cycle and hormone-induced lactogenesis in mammary explants has permitted an assessment of similarity of gene expression at the transcriptional level. Global transcriptome analysis using exact Wilcoxon signed-rank test with continuity correction and hierarchical clustering of Spearman coefficient demonstrated that hormone-induced mammary explants behave differently to mammary glands at secretory differentiation. Our results demonstrated that the mammary explant culture model mimics *in vivo* glands in immediate responses, such as hormone-responsive gene transcription, but generally did not mimic responses to prolonged hormonal stimulus, such as the extensive development of secretory pathways and immune responses normally associated with lactating mammary tissue. Hence, although the explant model is useful to study the immediate effects of stimulating secretory differentiation in mammary glands, it is unlikely to be suitable for the study of secretory activation.

## Declaration

This is to certify that

- (i) the thesis comprises only my original work towards the PhD except where indicated in the Preface,
- (ii) due acknowledgement has been made in the text to all other material used,
- (iii) the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

A handwritten signature in black ink, consisting of three Chinese characters: 林漢忠 (Lin Han Zhong).

---

Maurice Han Tong LING, BSc(Hons)  
Department of Zoology  
The University of Melbourne  
Australia  
SEN: 139520

## **Preface**

The background literature related to this thesis had been reviewed to March 2009.

Data chapters in this thesis have been written as a series of five papers (Chapters 2 - 6) which has resulted in repetition in the introductions and discussions of chapters on similar topics. At the time of printing, Chapter 2 has been published in Lecture Notes in Bioinformatics, volume 4774; Chapters 3 and 4 have been published in The Python Papers. These papers were jointly authored by Kevin Nicholas, Christophe Lefevre but in each case, I am the first author. All work described in this thesis were performed by myself except the planning and conduct of the microarray experiments on mouse mammary explants described in Chapter 5.

This research was funded by The University of Melbourne's Melbourne International Fee Remission Scholarship (MIRFS), Science Faculty Scholarship, and Cooperative Research Centre for Innovative Dairy Product's PhD studentship. Technical and computing resources were generously contributed in-kind by Victorian Bioinformatics Consortium, Monash University, Australia; High Performance Computing Unit, The University of Melbourne, Australia; Institute of Medical Informatics, National Yang Ming University, Taiwan; Bioinformatics Research Centre, Nanyang Technological University, Singapore. Conference travel to present results from this thesis was generously funded by The University of Melbourne's Postgraduate Overseas Research Experience Scholarship (PORES), Melbourne Abroad Travelling Scholarship (MATS), F.H. Drummond Travel Award, Department of Zoology at The University of Melbourne, and Cooperative Research Centre for Innovative Dairy Product.

## Acknowledgements

Paraphrasing from an old saying “It takes a village to raise a child; it takes a community to write a thesis”. It is without doubt in my mind that I am indebted to many people.

First and foremost, I will like to express my deep gratitude my PhD supervisors, Dr. Kevin Nicholas, Dr. Christophe Lefevre and Dr. Andrew Lonie, of The University of Melbourne, and Associate Professor Lin Feng, of Nanyang Technological University. I cannot imagine a better mentor and friend for my PhD than Kevin, for the dedication and interest in my project, and certainly for all your support and encouragements during my difficult times. For Christophe, your valuable suggestions and challenges had encouraged me to grow intellectually and yet providing a firm support. I extend many thanks to Andrew and Feng for your constructive criticisms and valuable suggestions to point me in a correct direction. This thesis would never been completed without your encouragement, guidance and admirable patience throughout the years.

I would like to acknowledge the computing support provided by the Victorian Bioinformatics Consortium, Monash University; Institute of Biomedical Informatics, National Yang Ming University, Taiwan; and High-Performance Computing Unit, The University of Melbourne. This work would not have been possible without these in-kind gestures.

I wish to thank the CRC for Innovative Dairy Products and The University of Melbourne for providing financial support for this work. The generous scholarships and training grants had given me the chance to present my work in multiple conferences and the chance to grow professionally.

Numerous constructive criticisms and suggestions from many people had gone into the preparation of this thesis: Associate Professor Peter Thompson and Professor Frank Nicholas of University of Sydney, for your insights into statistical modelling and microarray analysis; Professor Thomas Rindflesch of NIH and Professor Jonathan Wren of University of Oklahoma, for your taking time and effort for an informal review and critic of my text analysis system; Dr. Hugo Liu of MIT, for your guidance on using MontyLingua; Professor Hsu Chunnan and Professor Hsu Wenlian of Academia Sinica in Taiwan, for your insights into text mining; Dr. Jeffrey Chang of Duke University, for your assistance in abbreviation recognition and your kind gesture to allow me to replicate your system locally. Thank you very much.

The Department of Zoology and the Nicholas' group has provided me with marvelous working environment to complete my PhD. It is a privilege to work along-side with many of the staff and students. I wish everyone the best in the future endeavours. My sincerest thanks are extended to Phil Au, Joly Kwek and Edwin Wong, for your endless support and friendship had given me the energy to push on when I had doubted myself; Sonia Mailer, Dr. Mary Familiari, Josie D'Alessandro, for your encouragement along every step that I take; Professor David MacMillian, for letting me know that you are a fan of my work – that meant a lot to me.

I am deeply thankful for the constant love, support, patience and at times, blind faith, my friends and my team of Resident Advisers in University College, especially Genevieve Leach, have showered upon me to get me through all the trials and



tribulations that I have to face. You have given me a home away from home. Your love and support told me, silently but insistently, that I am never alone on this track.

I am deeply grateful to my parents and my family who had supported me silently but truly and concretely over these years overseas. Any of this would not have been possible without your support. To my brother, Melvin, affectionally known as my 'cat', thank you for tolerating my irritations, before and ever. Sorry for letting you go through your teenage years on your own, maybe that is for your best.

Perhaps my only regret is not able to finish this thesis fast enough for my grandmother to witness my graduation. She had passed on with dignity by removing her own oxygen supply on the afternoon of 22<sup>nd</sup> June 2008, 8 months after being diagnosed with terminal breast cancer caused by MAP kinase mutation in the insulin signalling pathway – a subject that I know intimately from this work. This thesis is for you.

# Table of Contents

Abstract.....	i
Declaration.....	iv
Preface.....	v
Acknowledgement.....	vi

## Chapter One: Introduction and Literature Review

1.1. Mammary Gland Development and Lactation.....	2
1.2. Mammary Explant Culture Studies to Determine the Role of Insulin, Prolactin and Glucocorticoid in Lactogenesis.....	6
1.3. Microarray Analyses of Mammary Gland Development and Lactation Cycle.....	8
1.4. Analysis of Gene Expression Microarray Results.....	10
1.5. Biomedical Literature Analysis.....	13
1.5.1. Brief History of Biomedical Literature Analysis.....	14
1.5.2. Current Areas of Research.....	16
1.5.3. Information Retrieval: Finding the Papers.....	17
1.5.3.1. Brief Descriptions of Information Retrieval Systems.....	18
1.5.4. Abbreviation Recognition: Aliasing the Names.....	19
1.5.5. Named Entity Recognition: Identifying the Players.....	20
1.5.6. Information Extraction: Getting the Facts.....	23
1.5.6.1. Co-occurrence.....	23
1.5.6.1.1. Brief Descriptions of Co-Occurrence Systems.....	24
1.5.6.2. Natural Language Processing / Template Matching.....	25
1.5.6.2.1. Brief Descriptions of Natural Language Processing / Template Matching Systems.....	27
1.5.6.3. Applications of Biomedical Information Extraction.....	31
1.5.7. Text Mining: Finding Hypotheses.....	34
1.5.8. Related Areas of Importance.....	35
1.5.8.1. Corpora.....	36
1.5.8.2. Databases.....	37
1.5.8.3. Evaluation Strategies.....	38
1.5.9. Challenges of Biomedical Literature Analysis.....	40
1.6. Microarray Analyses Assisted by Biomedical Literature Analyses.....	44
1.7. Tools for Visualizing Large Graphs.....	45
1.8. Objectives and Organization of this Thesis .....	46

## Chapter Two: Reconstruction of Protein-Protein Interaction Pathways by Mining Subject-Verb-Objects Intermediates

2.1 Introduction.....	48
2.2 System Description.....	49
2.2.1 Entity Normalization.....	50
2.2.2 Text Analysis.....	51
2.2.3 Protein-Protein Binding Finding.....	55
2.3 Experimental Results.....	56
2.3.1 Benchmarking Muscorian Performance.....	56
2.3.2 Verifying Protein-Protein Binding Interactions.....	57

2.3.3 Large Scale Mining of Protein-Protein Binding Interactions.....	58
2.3.4 Pilot Study - Protein-Protein Activation Interactions.....	59
2.4 Discussion.....	60

### **Chapter 3: Parts-of-Speech Tagger Errors Do Not Necessarily Degrade Accuracy in Extracting Information from Biomedical Text**

3.1 Introduction.....	64
3.2 Methods.....	65
3.2.1 Evaluating POS Tagging and Information Extraction Performance.....	65
3.2.2 Analysis of POS Tagging Errors.....	67
3.3 Results .....	68
3.3.1 Evaluating POS Tagging and Information Extraction Performance.....	68
3.3.2 Analysis of POS Tagging Errors.....	68
3.4 Discussion .....	74
3.5 Conclusions .....	78

### **Chapter 4: Filtering Microarray Correlations by Statistical Literature Analysis Yields Potential Hypotheses for Lactation Research**

4.1 Introduction .....	80
4.2 Methods.....	84
4.2.1 Microarray Datasets.....	84
4.2.2 Co-Occurrence Calculations.....	84
4.2.3 Comparing Co-Occurrence and Text Processing.....	85
4.2.4 Mapping Co-Expression Networks onto Text-Mined Networks.....	86
4.3 Results .....	86
4.3.1 Comparing Co-Occurrence Calculation Methods.....	86
4.3.2 Comparison of Natural Language Processing and Co-Occurrence.....	87
4.3.3 Mapping Co-Expression Networks onto Text-Mined Networks.....	88
4.4 Discussion .....	90
4.5 Conclusions .....	95

### **Chapter 5: The transcriptomics of lactogenesis in the murine mammary explant model; comparison with in vivo data**

5.1 Introduction.....	96
5.2 Materials and Methods.....	99
5.2.1 Mice.....	99
5.2.2 Tissue Culture.....	99
5.2.3 Microarray – Hybridisation and Analysis .....	100
5.2.4 Microarray Datasets.....	100
5.2.5 Assessing the Mammary Explant Model with Marker Genes.....	100
5.2.6 Comparison by Statistical Analysis and Hierarchical Clustering of Transcriptomics Data.....	101
5.2.7 Comparison by Gene Ontological Clustering.....	101
5.3 Results.....	102
5.3.1 Assessing the Explant Model with Marker Genes.....	102
5.3.2 Comparison by Statistical Analysis and Hierarchical Clustering of Transcriptomics Data.....	104
5.3.3 Comparison by Gene Ontological Clustering.....	105
5.4 Discussion.....	108

## **Chapter 6: General Discussion**

6.1 Introduction .....	115
6.2 Mouse Mammary Explant Model is a Suitable Model to Study Secretory Differentiation but not Secretory Activation.....	116
6.3 Computational Linguistics can Annotate Statistical Linguistics.....	117
6.4 Linguistic Analysis can Filter Microarray Data for Potentially Novel Hypotheses.....	119
6.5 Changing Face of Biological Research.....	121
6.6 Summary.....	122
References.....	123

## **Appendix A: Selected Source Codes**

A.1. Introduction.....	173
A.2. High-Level Description of the Codes.....	174
A.3. Selected Source Codes from Muscorian.....	175
A.4. Other Selected Source Codes.....	233

## List of Figures

- Figure 1a. Mammary gland development in mouse
- Figure 1b. General scheme of analyzing microarray data
- Figure 1c. Stages of biomedical literature processing showing examples of possible output from each stage
- Figure 1d. The funnel of knowledge-based information
- Figure 2a. Flow diagram illustrating the major operations of the process pipeline
- Figure 2b. Preliminary protein binding network of CREB
- Figure 2c. Preliminary protein binding network of insulin receptor
- Figure 2d. Preliminary protein activation network of insulin
- Figure 3a. Flowchart of evaluation procedure for Muscorian with native MontyLingua and MedPost-MontyLingua
- Figure 3b. Muscorian's generalization layer, from source text to subject-verb-object(s) structures
- Figure 4a. Percentage of correlation network analyzed from Master et al. (2002) are found in 1-mention PubGene co-occurrence
- Figure 5a. Hierarchical clustering of samples between in vitro and in vivo datasets using Spearman's coefficient

## List of Tables

- Table 1a. Summary of performances of biomedical literature analysis systems
- Table 2a. Penn Treebank Tag Set without punctuation tags
- Table 2b. Summary of the experimental results comparing the precision and recall measures
- Table 3a. Summary of Muscorian's performances evaluated using Learning Languages in Logic 2005 data
- Table 3b. Percentage breakdown of POS tags in MedPost corpus and errors in MontyTagger as percentage of POS tags assignation
- Table 3c. Penn Treebank Tag Set without punctuation tags
- Table 3d. Error breakdown of the six most common mis-assigned POS tags by MontyTagger on MedPost corpus
- Table 4a. Summary results of co-occurrence using PubGene or Poisson distribution
- Table 4b. Summary of incremental stepwise mapping of correlation coefficients from Master et al. (2002) to 1-PubGene co-occurrence network
- Table 5a. Summary of gene expression trends between IFP-induced mouse mammary explants and in vivo mammary glands using 5 classes of marker genes
- Table 5b. P-value table comparing pre- and post- lactogenic trigger in vitro and in vivo using exact Wilconox signed rank test with continuity correction
- Table 5c. Lists of Gene Ontologies that were common and single in IFP-up and LACT-up

## Publications and Presentations

### Peer-Reviewed Publications

**Ling, MHT**, Lefevre, C, Nicholas, KR, Lin, F. 2007. Re-construction of Protein-Protein Interaction Pathways by Mining Subject-Verb-Objects Intermediates. In J.C. Ragapakse, B. Schmidt, and G. Volkert (Eds.), Proceedings of the Second IAPR Workshop on Pattern Recognition in Bioinformatics (PRIB 2007). Lecture Notes in Bioinformatics 4774. (pp. 286-299) Springer-Verlag.

**Ling, MHT**, Lefevre, C, Nicholas, KR. 2008. Parts-of-Speech Tagger Errors Do Not Necessarily Degrade Accuracy in Extracting Information from Biomedical Text. The Python Papers 3 (1): 65-80

**Ling, MHT**, Lefevre, C, Nicholas, KR. 2008. Filtering Microarray Correlations by Statistical Literature Analysis Yields Potential Hypotheses for Lactation Research. The Python Papers 3(3): 4.

### Posters

**Ling, MHT**, Lefevre, C, and Nicholas, KR. 2006. A Pipeline for Analysis of Published Abstracts for Information on Protein-Protein Inter-Relations. Proceedings of the Fourth Asia-Pacific Bioinformatics Conference.

### Presentations

**Ling, MHT**. 2005. Molecular Pathways from Text. Engineering and Science Seminar 2005, The University of Melbourne.

**Ling, MHT**. 2006. Enhancing microarray analyses with text analyses. Cooperative Research Centre for Innovative Dairy Products Annual Meeting.

**Ling, MHT**. 2006. Functional modeling of mouse lactation using published abstracts and text mining. Language Technology Seminar Series. Department of Computer Science and Software Engineering, The University of Melbourne, Australia.

**Ling, MHT**. 2007. Re-constructing protein-protein interaction networks using text mining (for understanding microarray analyses). Institute of Information Science, Academia Sinica, Taiwan (R.O.C).

**Ling, MHT**. 2008. A tour of bioinformatics. Freshman Seminars 2008 – A Guided Tour in Science. Faculty of Science, National University of Singapore.





# Chapter One

## Introduction and Literature Review

Milk is the primary source of nutrition for the sustenance of a suckled newborn mammal. Hence, the process of synthesizing and secreting milk during lactation, is a critical process that crucially impacts on the survival of the young. The organ solely responsible for the synthesis and secretion of milk is the mammary gland.

The process of mammary tissue proliferation and differentiation during pregnancy, the onset and continued milk synthesis and secretion during lactation, the ceasing of milk synthesis and secretion, and apoptosis of the mammary tissue to return to pre-pregnancy state is known as the lactation cycle (Master et al., 2002) as shown in Figure 1a. The lactation cycle is complex in terms of developmentally-regulated changes in cellular organization and function (Master et al., 2002) involving the concomitant interactions of many hormones and growth factors, extracellular matrix and milk factors (Kumar et al., 2000)

Lactogenesis can be divided into 2 phases, lactogenesis I and lactogenesis II. Lactogenesis I, also known as secretory differentiation, the time when the mammary tissue initiate synthesis and secretion of milk (Hartmann, 1973). Lactogenesis II is the onset of copious milk secretion after parturition (Neville and Morton, 2001) and is followed by lactation, formally known as galactopoiesis, the continuation of copious milk production and secretion (Pang and Hartmann, 2007). Lactogenesis II and lactation are also known as secretory activation (Anderson et al., 2007).

Rodents such as mouse and rat are commonly used in the study of lactation cycle due to ease of animal housing compared to larger mammals. As a result, a large number of studies on lactation cycle have been done using rodents explant cultures (Bolander et al., 1981; Nicholas et al., 1983; Sakakura et al., 1987; Aoki et al., 1999, Zhao et al., 2002; Berlato and Doppler, 2009). In addition, there are microarray studies which have

analysed the global gene expression in the mammary gland during mouse lactation cycle (Master et al., 2002; Clarkson and Watson, 2003; Stein et al., 2003; Rudolph et al., 2007). Hence, there is a large amount of research data on rodent lactation cycle. Therefore, this thesis will be focused on mouse lactation and this review will concentrate on the rodent lactation cycle.

### ***1.1. Mammary Gland Development and Lactation***

With reference to Figure 1a, mouse mammary gland tissue develops dimorphically from the mammary crest from about Day 11 of mouse foetal development (Topper and Freeman, 1980). Further growth of the mammary crest is limited until late in foetal development when rapid epithelial growth occurs in females to form a mammary cord which opens to the eventual nipple. In contrast, the epithelial growth is not observed in males. Instead, the primitive mammary duct ruptures near Day 14 of gestation (Kratochwil and Schwartz, 1976) and is isolated in subepithelial mesenchyme (Topper and Freeman, 1980). The mammary cord remains dormant in females (Topper and Freeman, 1980) until stimulated by a combination of hormones at sexual maturation (Ball, 1998; Briskin et al., 1999; Hennighausen and Robinson, 2001; Vonderhaar, 1988) when ductal branching from the nipple is seen throughout the mammary fat-pad (Couldrey et al., 2002; Walden et al., 1998) and remains a minimal distance of 0.25mm apart (Faulkin and Deome, 1960).

During pregnancy, growth constraints between each duct are lifted in preparation for lactation and lobulo-alveolar structures are observed throughout the fat pad (Desjardins et al., 1968; Keough and Wood, 1979). Lactogenesis and lactation had been shown to be regulated by hormones as the level progesterone in the serum declines, accompanied by the elevation of prolactin level, was observed before parturition (Nicholas and Hartmann, 1981a; Topper and Freeman, 1980). Suckling of newborn pup provides the neuronal stimulation which triggers the release of oxytocin from the neurons located in the posterior pituitary gland resulting in milk letdown by the mammary gland (Jirikowski, 1992). A number of hormones had been suggested to form the lactogenic trigger (Topper and Freeman, 1980). This includes the withdrawal of progesterone before parturition (Kuhn, 1969) as progesterone was shown to inhibit lactation (Nishikawa, 1994). This is followed by the induction of prolactational hormones after

parturition (Topper and Freeman, 1980) such as prolactin (P), insulin (I), and glucocorticoid (F). This suggests that the lactogenesis II in the mouse is the result of a combination of stimuli by the neuronal and hormonal systems. A number of studies (Cowie et al., 1960; Cowie and Tindal, 1961; Nicholas and Hartmann 1981b) using organ-ablated rats such as Caesarean-sectioned, ovariectomized or ovariectomized rats to show the loss of lactation, despite the presence of suckling, confirmed the essential role of hormones in lactogenesis II.

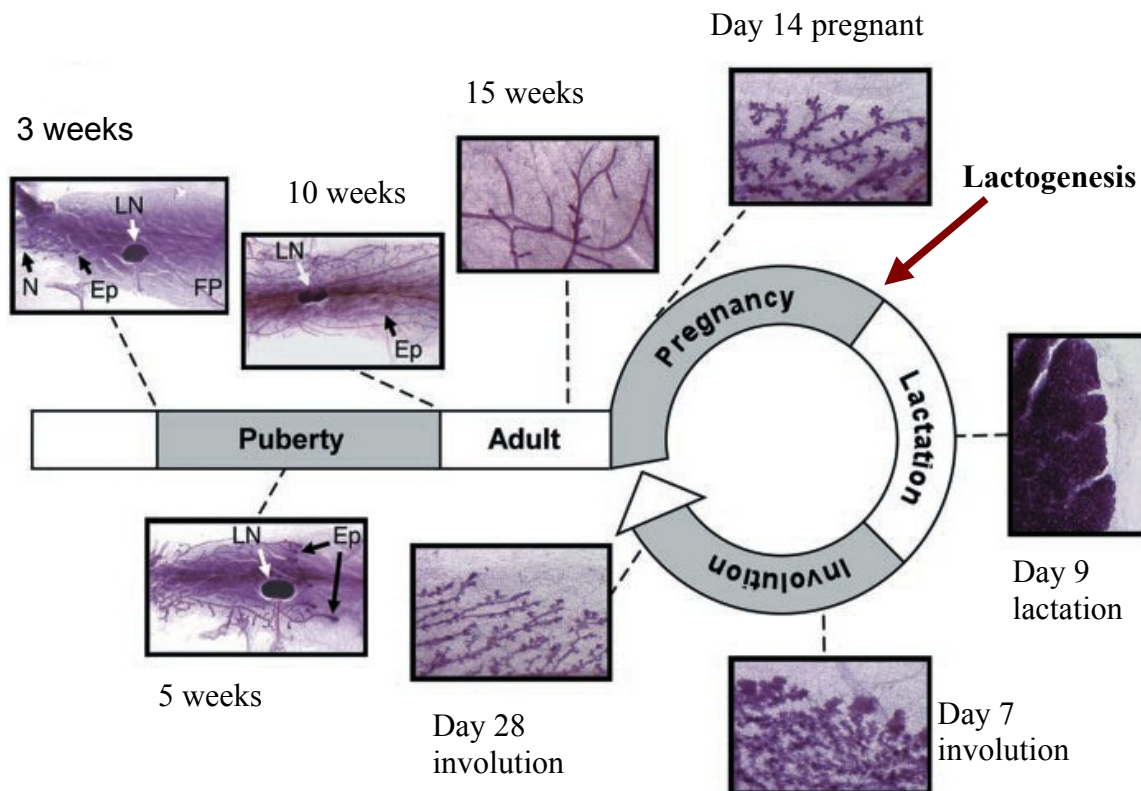


Figure 1a. Mammary Gland Development in Mouse. Whole mounts of murine mammary gland are shown to demonstrate the dramatic growth of the epithelial ductal tree during puberty and of the alveolar epithelial compartment during pregnancy. After lactation and postlactational involution, the gland returns to a state resembling the adult nulliparous gland. Glands shown were harvested at the onset of puberty (3 weeks), 5 weeks of age (mid-puberty), 10 weeks of age (completion of puberty), 15 weeks of age (adult nulliparous), day 14 pregnancy, day 9 lactation, day 7 involution, and day 28 involution. The nipple (N), epithelial tree (Ep), fat pad (FP), and lymph node (LN) are indicated. (Adapted from Master et al., 2002)

Prolactin was shown to be present at a high level in serum throughout lactation (Hart, 1972) and was suggested as a major regulator of lactation in a number of species (Fulkerson et al., 1975; Sinha et al., 1974; Weinstein et al., 1976). Prolactin binds to a membrane-bound receptor which is positively regulated by prolactin at physiological

levels (Bohnet et al., 1977; Djiane et al., 1979) and activates two proteins (Majumder and Turkington, 1971). These proteins were subsequently known to be Signal Transducer and Activator of Transcription-5a (Stat5a) (Bole-Feysot et al., 1998; Groner, 2002; Yamashita et al., 2001) and Janus Kinase 2 (JAK2) (Bole-Feysot et al., 1998; Groner, 2002; Liu et al., 1997). In a separate study, Stat5 was shown to be important for milk protein gene expression (Li and Rosen, 1995) which could explain part of prolactin signalling. In addition, prolactin had been shown to activate insulin signalling pathways, Protein Kinase B (Bole-Feysot et al., 1998) and phosphoinositide-3-kinase pathway (Abell et al., 2005).

Prolactin signalling results in a large number of downstream effects in mammary epithelial cells, for example, activating the transcription of the  $\beta$ -casein gene and is involved in the stability of  $\beta$ -casein mRNA (Ganguly et al., 1980b; Kulski et al., 1983a). Stat5a dimerizes with Signal Transducer and Activator of Transcription-5b (Stat5b) and Stat5a/Stat5b heterodimer (Ahonen et al., 2002) activates several transcription factors required for the expression of mammary genes (Malewski et al., 2002; Zhao et al., 2002). The prolactin signaling pathway can be inhibited at two points in the mammary epithelial cells: protein tyrosine phosphatase 1B (PTP1B) can deactivate Stat5a/Stat5b in the cytosol (Aoki and Matsuda, 2000) and T-cell PTP1B (TC-PTP1B), which is expressed in mammary epithelial cells (Aoki et al., 1999), can specifically deactivate Stat5a/Stat5b in the nucleus by dephosphorylation (Aoki and Matsuda, 2002).

Glucocorticoids are membrane-permeable steroid hormones that bind to a cytoplasmic receptor, which is subsequently translocated into the nucleus (Bamberger et al., 1996). The hormone-bound glucocorticoid receptor had been shown to form a complex with Stat5 before binding to the casein gene (Groner, 2002). Prior to lactogenesis, glucocorticoid-bound receptor was found to be localized in the endoplasmic reticulum and golgi apparatus (Mills and Topper, 1969; Mills and Topper, 1970).

Although early studies indicated a role of insulin in casein gene expression (Juergens et al., 1965), elucidating the role of insulin in the process of mammary development and lactation had been confounded by the fact that a major role of insulin is glucose

homeostasis (Plum et al., 2006). During pregnancy, insulin was shown to be essential for the terminal differentiation of mammary epithelial cells (Berlato and Doppler, 2009). However, during the process of lactation, insulin has an important role in uptake of nutrients by the mammary gland (Bequette et al., 2001; Griinari et al., 1997; Mackie et al., 2000) while causing other tissues to be resistant to insulin by perpetual hyperinsulinemia during late pregnancy and early lactation (Bequette et al., 2002; Carrascosa et al., 1998). However, mammary tissues from virgin mice had been shown to be less sensitive to insulin compared to the mammary tissues from pregnant or parous mice (Bolander, 1983) suggesting that pregnancy is required for the insulin sensitivity in the mammary glands. A further study by Bolander (1984) demonstrated that injecting progesterone into virgin mice resulted in comparable insulin sensitivity of the mammary glands to that of pregnant or parous mice suggesting that progesterone is crucial for insulin sensitivity of mouse mammary glands. This effect had also been shown by Bolander (1984) to sustain up to 4 weeks after the removal of progesterone, suggesting that the ability of mouse mammary tissue to maintain sensitivity to insulin during lactation can attributed to the effect of progesterone during pregnancy. In addition, insulin has a role in making mammary cells more responsive to other hormones such as growth hormone (Butler et al., 2003). In vivo examination of lactating and non-lactating rats showed that the level of circulating insulin in lactating rats is lower than non-lactating rats, suggesting that lactation is hypo-insulinemic process (Burnol et al., 1986). This observation was supported by a study using diabetic rats suggesting that short-term insulin deficiency does not prevent the secretion of milk components (Kyriakou and Kuhn, 1973). This evidence seems to suggest that the level of insulin required in circulation varies at each stage of mammary gland development and the lactation cycle. A possible explanation for the reduced insulin requirement post-partum compared to pre-partum is the activation of insulin pathways by high level of prolactin (Abell et al., 2005) in serum throughout lactation (Hart, 1972).

After weaning, the mammary lobulo-alveolar duct system undergoes involution, characterized by apoptosis and tissue remodelling, which returns the glands to a near pre-pregnancy state (Accorsi et al., 2002; Baik et al., 1998; Tatarczuch et al., 1997), as shown in Figure 1a.

## ***1.2. Mammary Explant Culture Studies to Determine the Role of Insulin, Prolactin and Glucocorticoid in Lactogenesis***

Mammary explant culture was developed in the 1950s by Elias and colleague (Elias, 1957; Elias and Riveria, 1959) to study the effects of hormones on cultured mammary tissues. Insulin was routinely added into the culture media (Elias, 1957; Elias and Riveria, 1959) for the maintenance of cell viability (Barnawell, 1965). Experimental studies on the role of hormones in lactogenesis I using whole animals will require invasive surgeries and overt disruptions to the physiological hormone balance (Cowie, 1960; Nicholas and Hartmann, 1981b). Hence, mammary explant culture model provides ease of experimental manipulation compared to whole animal studies. In contrast to primary monolayer cell culture, mammary explants retained the native extracellular matrix induced 3-dimensional tissue structure of the mammary gland which had been shown to be required for the synthesis of milk proteins (Aggeler et al., 1988).

Using the explant culture and casein genes as phenotypic markers of lactation, Juergens et al., (1965) demonstrated that insulin (I), prolactin (P) and glucocorticoid (F) were needed for casein synthesis. Juergens et al. (1965) measured the incorporation of radio-labelled phosphate into casein in mouse mammary explants cultured in Medium 199 without hormone supplementation and Medium 199 supplemented with IFP, singly or in combination. This experiment found that only explants cultured in media supplemented with all 3 hormones showed increased radioactivity in casein transcripts; thus, implicating the role of IFP in lactation. However, the individual roles of each of these 3 hormones was not known and was a subject of subsequent studies.

A series of elegant mammary explant experiments using tissue from different animals demonstrated that prolactin was needed for milk protein gene expression (Devinoy et al., 1978; Terry et al., 1977; Turkington et al., 1965; Turkington et al., 1967). Most importantly, Matusik and Rosen (1978) cultured mammary explants in IF-supplemented media before transferring the culture into IFP-supplemented or IP-supplemented media and demonstrated the requirement of prolactin, in the presence of insulin, for the expression of casein genes. Prolactin had been suggested to play a role in sensitizing the mammary epithelial cells to other hormones (Oka and Topper, 1972). Matusik and Rosen (1978) also suggested that glucocorticoid is unnecessary for the transcription of

casein genes as casein genes were expressed in IP-supplemented media. However, Bolander et al. (1979) demonstrated that glucocorticoid was retained in the mammary fat pad following culture in IF-supplemented media suggesting that glucocorticoid was carried over from IF-supplemented media to IP-supplemented media in Matusik and Rosen (1978). Therefore, particular care is required for the experimental design to prevent retention of glucocorticoid in the fat pad (Bolander et al., 1979). However, retention of glucocorticoid in the mammary fat pad implied that Matusik and Rosen (1978) could not be definitive on the role of prolactin as an absolute requirement for casein gene expression as elevated casein gene expression might also be observed if the explants had been cultured in FP-supplemented media before transferring to IP-supplemented media.

Using the mammary explant model, glucocorticoid was found to play a role in the transcription of  $\beta$ -casein mRNA (Chomczynski et al., 1986; Ganguly et al., 1979; Kulski et al., 1983b; Nagaiah et al., 1981) through Oct-1 transcription factor (Dong and Zhao, 2007; Dong et al., 2009) in the presence of prolactin and insulin (Ganguly et al., 1980b). Bolander et al. (1981) measuring the RNA concentration of  $\beta$ -casein in rough endoplasmic reticulum also determined that a combination of prolactin, insulin and glucocorticoid is necessary for the accumulation of  $\beta$ -casein transcripts. It has been suggested that glucocorticoid has a role in determining the half-life of milk protein gene mRNA (Chomczynski et al., 1986; Poyet et al., 1989).

Insulin had been shown to be essential in the transcription of the  $\beta$ -casein gene (Bolander et al., 1981; Chomczynski et al., 1986; Kulski et al., 1983a; Topper et al., 1984a; Topper et al., 1984b) and the stability of  $\beta$ -casein transcript by lengthening its poly(A) tail (Choi et al., 2004) in the presence of prolactin. The view of insulin playing a “permissive” role in maintaining cell viability in vitro (Elias, 1957) was refuted by Nicholas et al. (1983) who demonstrated that casein cannot be induced in the absence of insulin. Other growth hormones such as epidermal growth factor, nerve growth factor, platelet-derived growth factor, fibroblast growth factor, and even proinsulin could not substitute the role of insulin in casein synthesis (Nicholas et al., 1983). A recent study demonstrating the ability of explants to respond to IFP by expressing milk protein genes after 10 days of culture without insulin or other hormones supported the view that

insulin is not simply required for cell viability and is directly involved in the expression of milk protein genes (Brennan et al., 2008).

Although a large number of explant studies (Chomczynski et al., 1986; Ganguly et al., 1979; Kulski et al., 1983b; Nagaiah et al., 1981; Topper et al., 1984a; Topper et al., 1984b) using mammary explant culture to demonstrate IFP as the minimal set of hormones for the expression of  $\beta$ -casein, Collier et al. (1977) demonstrated that IFP-induced explants might be limited in secretory ability, suggesting different hormonal requirements for the initiation of milk synthesis and the initiation of milk secretion which corresponds to our current definition of secretory differentiation (Hartmann, 1973) and secretory activation (Neville and Morton, 2001) respectively. This suggested that despite evidence the ability of IFP-induced mouse mammary explants to express milk protein genes such as  $\beta$ -casein which can be used as marker genes for lactogenesis I, the IFP-induced mouse mammary explant culture model had not been shown to be a suitable in vitro model for a more comprehensive study of lactogenesis I.

### ***1.3. Microarray Analyses of Mammary Gland Development and Lactation Cycle***

Although the marker gene approach had shown that IFP is necessary for lactation, it is limited in the number of genes studied. Recent developments in technologies such as gene expression microarray have allowed thousands of genes to be analyzed in a single experiment. This technology has been applied to study the transcriptional profile of the mouse mammary gland at various stages of the development and lactation. To date, there are 4 main studies using microarray to examine different aspects of mouse lactation cycle.

The first study was conducted by Lewis Chodosh's group (Master et al., 2002). Thirteen Affymetrix microarrays were used to examine whether changes in the mammary transcriptional profile corresponded to the characterized mouse mammary gland development and lactation cycle. Using pair-wise Pearson's correlation coefficient across adjacent time-points, Master et al. (2002) demonstrated that different sets of genes are up-regulated at different phases of the mouse mammary gland development and lactation cycle from the onset of sexual maturity to pregnancy, and from early



lactation to involution (Figure 1a).

The microarray analysis undertaken by Clarkson and Watson (2003) focused on the process of involution rather than the entire lactation cycle. They chose 12 time-points across the entire lactation cycle with 2 replicates each (a total of 24 microarrays). However, 5 of the time-points (12 hours, 24 hours, 2 days, 3 days and 4 days) were used to examine the early stages of involution after forced weaning at day 10 lactation. Among the genes that were induced during involution were those involved in inflammation response. Thus, Clarkson and Watson (2003) suggested that innate immune responses such as complement activation in the mammary tissue as a result of inflammation is the initiation of involution. This also co-incides with the activation of neutrophils into the mammary gland at the initiation of involution (Clarkson and Watson, 2003)

Stein et al. (2004) examined the first stage (day 1, 2, 3 and 4) and second stage (day 20) of involution. Stein et al. (2004) corroborated the findings of Clarkson and Watson (2003) to show that the immune cascade and Stat3 were activated within the first 4 days post-weaning despite using different strains of mouse, C57/BL6 and Balb/C. Stein et al. (2004) identified distinct responses between the first and second stages of involution using both microarray analysis and histological examination. The first stage of involution was characterized by neutrophil activation while the second stage was characterized by macrophage and eosinophil activation. Both Stein et al. (2004) and Clarkson and Watson (2003) demonstrated that immune cells are involved in the process of involution and macrophages may be involved in the removal of apoptotic cell during the second stage of involution (Stein et al., 2004). This suggests that in vitro study of involution using mammary explant culture is unlikely to be comprehensive.

Rudolph et al. (2007) used 29 microarrays to study lipid synthesis in the mouse mammary gland following diets of various fat content from 8% to 40% as the lactating mouse needs to synthesize and secrete an equivalent of its own body weight in fat during the 20-day lactation. This study showed that genes encoding for nutrient transporters into the cell are up-regulated following increased food intake and genes encoding for the lactose synthesis pathway are up-regulated 5 to 15 fold between day 17

pregnancy and day 2 lactation. This corresponded to the increase in lactose synthesis during parturition (Nicholas and Hartmann, 1981b). Rudolph et al. (2007) also found that adaptations to various dietary fat content occurred primarily in the liver whereas the mammary gland was primarily responsible for uptake of nutrients and the synthesis and secretion of milk components. This demonstrated that the regulation of lipid and lactose synthesis in the mouse mammary gland occurred primarily at the transcriptome level.

Collectively, the studies discussed above have shown that the cyclic nature of mammary gland development during pregnancy and lactation is reflected at the transcriptional level, suggesting that gene expression microarray can be a useful tool to study various aspects of mouse lactation. Although the studies discussed above had not focused on lactogenesis I and II, each study consisted of time-points before and after parturition. In addition, there is no study examining the transcriptional profile of mouse mammary explants before and after IFP induction. Microarrays allow the comparison of gene expression in mouse mammary explants before and after IFP induction and that of in vivo mouse mammary glands before and after parturition to address the question of whether the IFP-induced mammary explants mimic the in vivo lactogenic response.

#### ***1.4. Analysis of Gene Expression Microarray Results***

A large area of mathematical statistics and informatics has been directed to support microarray experimental data analysis. Before analysis of gene expression (Figure 1b), image scans from each microarray slide must be normalized and pre-processed for anomalies, such as missing values (Herrero et al., 2003). This is done within a slide to remove intra-array variations and across array to remove inter-array variations (Schadt et al., 2001; Tseng et al., 2001; Yang et al., 2002).

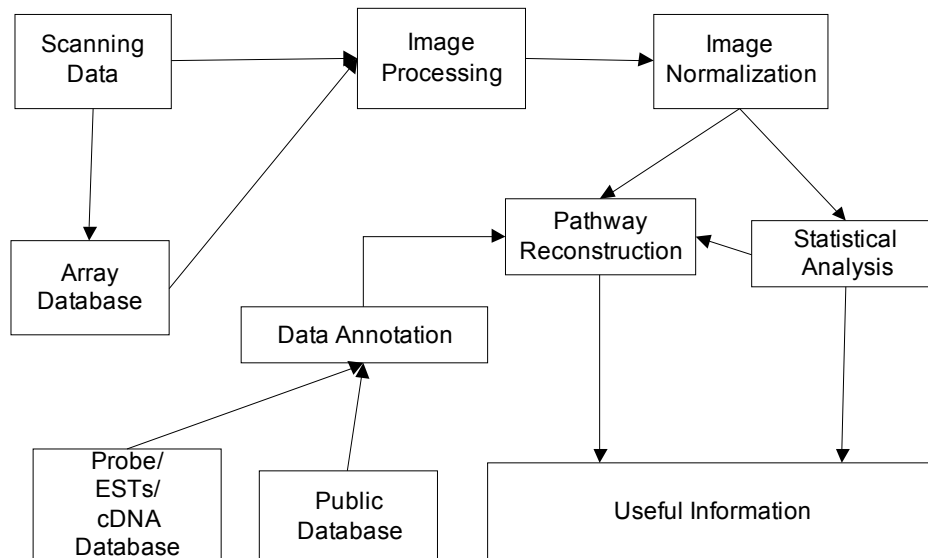


Figure 1b. General Scheme of Analyzing Microarray Data. Raw scanned images can be stored into a database or processed immediately. Image processing includes assembling repeated scans and format conversion whereas image normalizing adjusts image intensities, inter-chip and intra-chip, by using a series of standards built onto the microarray chip. Image processing and normalization forms the data preprocessing step. Information about the probe/ESTs/cDNAs embedded on the chip are annotated by referencing it to public databases. This knowledge will then be superimposed on the normalized image, giving each spot an identity. Only then, can it be further processed by either statistical analysis or pathway reconstruction approaches.

After data pre-processing, actual analysis can proceed using a combination of statistical and pathway reconstruction approaches. Statistical means includes classical statistics, such as, Student's t-test (Baldi and Long, 2001) and analysis of variance (Didier et al., 2002; Wu et al., 2007) and their derivatives (Broberg, 2003; Brown et al., 2001; Butte et al., 2001; Chapman et al., 2002; Chen et al., 2002; Churchill, 2004; Culhane et al., 2002; Fuhrman et al., 2000; Hastie et al., 2000; Holter et al., 2001; Holter et al., 2000; Hoon et al., 2002; Kerr and Churchill, 2001; Kroll and Wolfl, 2002; Li and Hong, 2001; Manduchi et al., 2000; Nueda et al., 2007; Pavlidis, 2003; Sasik et al., 2001; Taib, 2003; Wall et al., 2001; Xing and Karp, 2001; Zhou and Abagyan, 2002) or learning methods, such as, neural network (Bidaut and Stoeckert, 2009; Herrero et al., 2003; Lynn et al., 2009), self-organising map (Covell et al., 2003; Kasturi et al., 2003; Tamayo et al., 1999; Touya et al., 2008), support vector machines (Brown et al., 2000; Pavlidis et al., 2001; Pirooznia and Deng, 2006; Zhu et al., 2009), Bayesian network (Bhattacharjee et al., 2008; Chi et al., 2007; Hu and Qin, 2009; Rogers and Giromali, 2005; Chen et al.,

2006; Zou and Conzen, 2005; Kim et al., 2003) and Gaussian network (Shimamura et al., 2007; Toh and Horimoto, 2002; Li and Gui, 2006). Using these approaches, patterns of gene expression which may be present across treatments can be easily identified as co-expression profiles or maps (Li et al., 2006a). At the same time, it had been shown that genes with related functions exhibit similar co-expression profiles across different conditions (Kim et al., 2003; Reverter et al., 2005a; Hughes et al., 2000), within species (Nachman et al., 2004, Gardner et al., 2003, Guelzim et al., 2002) and across species (Stuart et al., 2003, Winter et al., 2004). Comparing co-expression maps of different treatments or cell types had yield important biological insights into human embryonic stem cells and embryoid bodies (Li et al., 2006a). Analyzing single co-expression map can also yield important biological results, such as identifying osmotic shock response network in yeast within the entire genomic co-expression map (Ulitsky and Shamir, 2007).

Co-expression maps could be used to infer gene networks from microarray data by essentially reverse engineering from the co-expression profiles to possible genetic regulatory networks (Lai, 2008; Thomas et al., 2004; Maraziotis et al., 2007; Margolin et al., 2006; Qiu et al., 2009; Rawool and Venkatesh, 2007). Recent work included inferring from single microarray data set (Toh and Horimoto, 2002), collection of data sets (Lee et al., 2004; Reverter et al., 2005b), single time series data set (Kim et al., 2003; Kimura et al., 2008; Maraziotis et al., 2007; Zou and Conzen, 2005), and collection of time series data set (Wang et al., 2006; Bickel et al., 2009). However, the accuracy of inferred gene networks is a current topic of study (Reverter et al., 2005b; Husmeier, 2003; Werhli and Husmeier, 2007; Zhao et al., 2008). A number of studies (Lee et al., 2004; Li et al., 2006a; Reverter et al., 2005b) had mapped the inferred gene networks or network clusters onto Gene Ontology alone (Ashburner et al., 2000) to evaluate the statistical confidence of Gene Ontology over-representation in these clusters as a measure of functional relevance of the inferred networks or the suitability of the clustering algorithms in individual cases (Datta and Datta, 2006).

The development of resources, such as ontologies, over the recent years had made microarray analysis more manageable. For example, inferred gene networks can be mapped onto Gene Ontology (Beissbarth, 2006) or MGED Ontology (Whetzel et al.,

2006) singly or an array of ontological, pathways and text resources (Montaner et al., 2006). Mapping gene networks onto ontologies had allowed for future statistical analysis with tools, such as Gostat (Beissbarth, 2006) and BiNGO (Maere et al., 2005). At the same time, it can allow for the deduction of functional gene sets for various medical conditions (Ovaska et al., 2008), such as basal cell carcinoma (O'Driscoll et al., 2006) and breast cancer (Liu et al., 2008).

Pathway reconstruction makes use of the readily available knowledge of metabolic or regulatory pathways and maps the genes expressed onto these known pathways, which will provide information as to the nature of pathways being active in each treatment. For example, Moran et al. (2007) identified a microglial gene regulation network that is activated by interferon-gamma by mapping microarray results to a database of eukaryotic molecular interactions. Okamoto et al. (2007) used phylogenetic profiles, microarray data, and KEGG to predict functional relationships of nitrogen-metabolism genes. Bickel et al. (2009) used time-course gene expressions analysis on time-lagged model based on the assumed rates of gene transcription and translation to construct a gene regulation network. This is in contrast with statistical analyzes which do not require prior knowledge of biochemical processes. Pathway reconstruction uses two different approaches, either regulatory pathways, such as, by promoter analysis, or metabolic pathways. Some of the techniques used in this area include, functional clustering (Boutros and Okey, 2005; Hawse et al., 2003; Juan and Huang, 2007; Kalish et al., 2004; Kwon et al., 2004; Lee et al., 2003; Pandey et al., 2004; Pusztai et al., 2003; Song et al., 2003; Tabibiazar et al., 2003; Yang et al., 2009), weight matrices (Weaver et al., 1999) and pathway scores (Zien et al., 2000; Presson et al., 2008), which provides increased biological relevance. Dense areas in pathway maps of interactions had been shown to correspond to known protein complexes or clusters (Bader and Hogue, 2003). Statistical analysis and pathway reconstruction are not mutually exclusive to each other (Wu et al., 2004) but can be used together to yield important results (Brady et al., 2006; Herzel et al., 2001).

### ***1.5. Biomedical Literature Analysis***

With rising emphasis in genomics, transcriptomics and proteomics from the end of the last century, the focus of biomedical research is shifting from the study of individual

proteins and genes to entire biological systems, such as tissues or whole organisms. Experimental techniques used for such analyses, like mass spectrometry and microarrays, often generate large data sets. A group of biologists must then collaborate to make sense of this large set of experimental data, and often requires connections with research areas outside their own core competencies, which exists as published literature in various research areas. This has resulted in a need to be versed in research areas other than the researcher's own specialty. In addition, the amount of information, in the form of published articles, is increasing exponentially, making it difficult for a researcher to keep abreast with relevant literature manually (Hunter and Cohen, 2006), even on specialized topics.

Due to these changes, literature processing tools are becoming essential to researchers (Cohen and Hersh, 2005) as it was estimated that only about 20% of biological knowledge exist in structured formats, such as in databases, while the remaining 80% are in natural language documents (He et al., 2009; Hunter and Cohen, 2006). They include targeting relevant papers, known as information retrieval; identifying gene or protein or chemical entities; identifying abbreviations; extracting facts from the literature, known as information extraction; and in some instances, generating hypotheses. This review shall briefly examine the historical roots and current state of biomedical literature analysis. The computational procedures of 18 systems will be briefly described to illustrate some of the current methods used, and its related areas of importance before defining the objectives and organization of this thesis.

### **1.5.1. Brief History of Biomedical Literature Analysis**

Don Swanson initiated interest in biomedical literature analysis by analyzing publications semi-automatically and suggested links between separate areas of research, such as fish oil and Raynaud's syndrome (Swanson, 1986), migraine and magnesium (Swanson, 1988) in the mid-1980s.

At around the same time, the First Message Understanding Conference (MUC-1) was held in 1987, which explored formats for recording information in documents. In 1989, MUC-2 concentrated on template filling of information and formulated the details of precision and recall measures, which is still in use today. MUC-3 (1991) and MUC-4

(1992) were centered on compiling and completing template from information in terrorist reports. Therefore, it had no direct bioinformatics relevance but resulted in improved techniques.

In 1992, the First Text Retrieval Conference (TREC-1) was initiated by the National Institute of Standards and Technology (NIST) and U.S. Department of Defense and used the idea of challenge evaluation tasks to tease out the state of the art of that time. Both TREC-1 and MUC-5 in 1993 were greatly influenced by the Tipster program (a U.S. Government program) which emphasized on evaluation-driven research (Prange, 1996), setting the tone for future conferences. TREC-2 in 1993 was critical for providing the baseline performance for main tasks, which was then expanded to having different tracks for various tasks in future TRECs. Each track in TREC was started based on interests and notably, a genomic track was started in 2003 (Hersh et al., 2004) with a subsequent TREC in 2006. In short, MUC and TREC conferences had significantly advanced the field of information retrieval and extraction by their challenge tasks due to rigorous use of systematic common evaluations (Hirschman, 1998).

The earliest work in text mining for genomics was by Timothy Leek (1997). Fukuda et al. (1998) pioneered protein name recognition (named entity recognition) in text in the 3<sup>rd</sup> Pacific Symposium on Biocomputing. Craven and Kumlein (1999) and Blaschke et al. (1999) independently published the first work on recognition of relationships between entities (proteins, genes, and small molecules). By 2000, the focus had shifted to the recognition of relationships between entities (proteins, genes, and small molecules), with Shatkay and Wilbur (2000) and GENIES (Friedman et al., 2001) as one of the first systems. Following GENIES, the field of biomedical literature analysis for information retrieval and information extraction was extremely active with numerous systems being developed (reviewed in later sections). The Pacific Symposium on Biocomputing between the years 2001 and 2004 included predominantly presentations on various aspects of biomedical literature analysis and other related conferences, such as Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), saw an increase in the number of similar presentations. Specific workshops for biomedical literature analysis, such as BioLink 2004, were run in that period. It could be said that the years between 1998 and 2004 were the golden age of

biomedical literature analysis.

By the end of 2004, the general emphasis of biomedical literature analysis had diminished at these conferences. This might be due to the asymptotic performance of various systems and a serious lack of benchmark data, such as biomedical corpora tagged for various purposes (Leser and Hakenberg, 2005). This might also mean that conventional means of analysis and technology transfer from more traditional fields, such as computational linguistics for understanding general text, had reached its maximum. The changes were more likely due to the rising interest in other fields, such as microarray and sequence analyses. The Fourth Asia Pacific Bioinformatics Conference (APBC 2006) saw more than half of the posters in the area of microarray and sequence analyses while only about 5% in biomedical literature analysis. However, the golden age of biomedical literature analysis of 1998 and 2004 had left us with preliminary resources and techniques that could be collated for other purposes.

Interestingly, although the field of biomedical literature analysis arose from Don Swanson's work in the mid-1980s (Swanson, 1986; Swanson, 1988), hypothesis generation (text mining) was not adopted into mainstream biomedical literature analysis. This suggested that the field of biomedical knowledge is still largely uncharted with gems for many explorers to find in the years to come. With that optimistic thought, we shall explore the current areas of research in biomedical literature analysis.

### **1.5.2. Current Areas of Research**

Although the primary utility of biomedical literature processing is obtaining the relevant research articles (information retrieval; IR), extracting facts (information extraction; IE), and in some instances, drawing new hypotheses (text mining; TM) as shown in Figure 1e., there are five concentrations of research efforts instead of the mentioned three. The other two are Named Entity Recognition (NER) and Abbreviation Recognition (AR), which are more domain specific (Cohen and Hersh, 2005). Comparatively, IR, IE and TM tend to be less domain-specific.

Before focusing on each of the core research areas, it is crucial to understand some commonly used performance measures. Systems are typically measured in terms of



precision (number of correct predictions divided by the total number of predictions) and recall (number of correct predictions divide by the total number of correct predictions in the test set) (Cohen and Hersh, 2005). Precision (P) and recall (R) can be combined into a single F-score, defined as the harmonic mean of precision and recall,  $2PR/(P+R)$  (Natarajan et al., 2005; Tsai et al., 2006). A more extensive treatment of evaluation strategies will be given in Section 1.8.3.

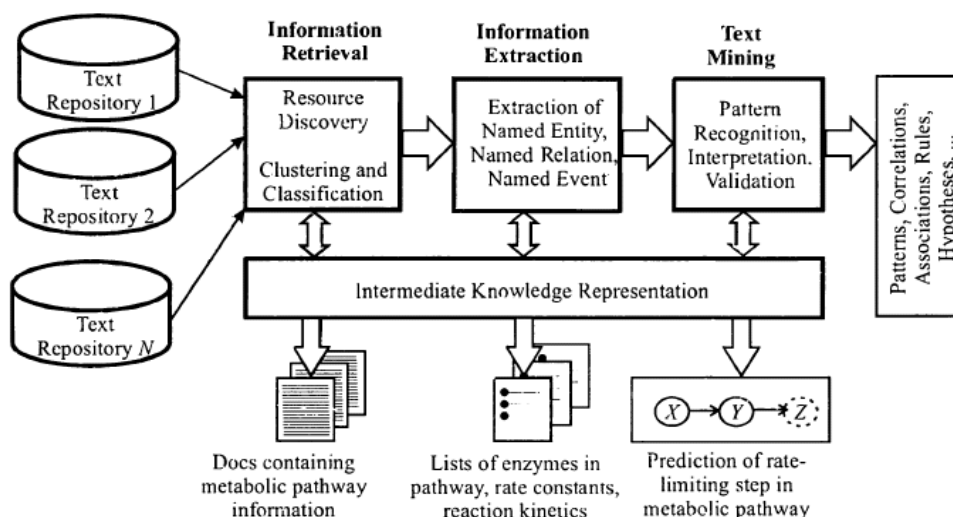


Figure 1c. Stages of biomedical literature processing showing examples of possible output from each stage. Adapted from (Natarajan et al., 2005)

### 1.5.3. Information Retrieval: Finding the Papers

Information retrieval (IR) tools aim to identify text (full text, abstracts, sentences) pertaining to a certain topic of interest, which might be user-defined (*ad hoc* IR) or pre-defined categories, as in text categorization (Han et al., 2006), or a hybrid of both (Chen et al., 2006b). The best-known biomedical IR system, PubMed, is an *ad hoc* IR system, which uses Boolean logic and a vector space model, and is available as a programmatic interface, PubMed EUtils.

Both Boolean logic and the vector space model are established IR methodologies (Gerard et al., 1983). Boolean logic builds on keyword queries where multiple keywords are chained using Boolean operators, such as “(mouse AND cytokines) NOT interferon”. In the vector space model, each document is represented by document-term

vector, calculated by a frequency-based weighing scheme, which is then compared to the query vector (Aronson et al., 2000; Wilbur and Yang, 1996). This had also been used for thematic analysis of the AIDS literature (Wilbur, 2002).

Building on PubMed's IR system, more advanced tools like Textpresso (Muller et al., 2004) and MedMiner (Tanabe et al., 1999) employed named entity recognition (see 1.3.2) to identify text for protein and gene names. Shatkay et al. (2008) used named entity and terms recognition before presenting the results to a machine-learned classifier to score biomedical text. Results from an IR system are usually presented as a long list which gives a poor overview. Recent developments had attempted to either summarize search results into other representations using Gene Ontology (Doms and Schroeder, 2005; Chiang et al., 2006) or to represent pair-wise searches as network (Simon et al., 2004), as it had been shown that biomedical ontologies can significantly enhance the quality of document clustering (Yoo et al., 2007) which can be used as input for text mining. Biological ontologies had also been used to restrict the amount of returned documents in PathBinderH (Ding et al., 2005) and to assist in document navigation (Baker et al., 2008). Query term matching (Aronson et al., 2000) and expansion (Aronson and Rindfleisch, 1997) is an important aspect and a current obstacle in biomedical IR due to multiple synonyms describing the same entity. For example, 'yeast', also known as 'baker's yeast', can be both '*Saccharomyces cerevisiae*' or '*Schizosaccharomyces pombe*' or their abbreviated forms, '*S. cerevisiae*' and '*S. pombe*' respectively. Although matching of query terms to biomedical vocabularies like Medical Subject Headings (Simon et al., 2004), UMLS (Aronson, 2001; Hersh et al., 2000) and OWL-DL ontology (Witte et al., 2007) is possible, the query term expansion is still largely an open problem (Jensen et al., 2006).

#### **1.5.3.1. Brief Descriptions of Information Retrieval Systems**

MedMiner (Tanabe et al., 1999) interrogates PubMed and GeneCard. User query of GeneCard database obtained a list of associated genes, which could be filtered with a user defined list of genes of interest and used to query PubMed. Results from PubMed were analyzed at sentence-level and only sentences with at least one gene and keyword remained. Final results were presented to the user after clustering based on rules or keywords.

Textpresso (Muller et al., 2004) is an example of a system able to extract multiple types of information from biomedical literature (Jensen et al., 2006). Text was POS tagged with Brill tagger (Brill, 1995) trained on *C. elegans* literature but was not further used. Instead, the same text was tagged by a set of 33 tags, forming the Textpresso Ontology, consisting of 14500 Regular Expressions. Exhaustive indexing on the ontology was carried out to facilitate text retrieval. A further work (Chen et al., 2006b) expanded on Textpresso by automatic clustering of the results into document categories using phrase clustering and support vector machines.

PathBinderH (Ding et al., 2005) intersected PubMed search results and NCBI Entrez Taxonomy. Both search results and taxonomical terms were user-defined. Only results that were indexed in the correct taxonomy were returned to the user.

Shatkay et al. (2008) used MedPost (Smith et al., 2004) and YamCha (Kudo and Matsumoto, 2000) to identify terms in the source text to train a classifier to identify the required articles.

#### **1.5.4. Abbreviation Recognition: Aliasing the Names**

Growth in biomedical terminology parallels the growth of biomedical literature and many of these biomedical terminologies have abbreviations. There are two foreseeable uses in collecting terminologies and their abbreviations. Firstly, abbreviations can aid information retrieval by providing a means to expand search terms to cover terminological variants that have the same abbreviation. Secondly, a dictionary of terminologies and abbreviation may facilitate text processing of multi-word terms as described in the previous section. Abbreviation Recognition (AR) is pair recognition of a terminology (may be a phrase or an entity) and its corresponding abbreviation from free text.

Most of the progress in AR was made between the years 2001 and 2004, and had mostly adopted rules-based techniques with a statistical score. Liu and Friedman (2003) were probably the only group that used mainly statistical co-location to determine abbreviations and their phrases. The main drawback of AR by statistics is the need for a

large collection of text, known as a corpus (corpora for plural).

Rules-based methods used the knowledge of how abbreviations were being formed and had been well described by Jeffrey Chang (Stanford University's Abbreviation Server) (Chang et al., 2002; Chang, 2003), which demonstrated 97% precision at 22% recall (F-score = 0.36) and 95% precision at 75% recall (F-score = 0.88). Yu et al. (2002b) (AbbRE) and Schwartz and Hearst (2003) both used pattern matching rules and achieved 96% precision with 70% recall (F-score = 0.81), and 96% precision with 82% recall respectively (F-score = 0.88). SaRAD system (Adar, 2004) reported 95% precision with 85% recall (F-score = 0.9). AcroMed (Pustejovsky et al., 2001) reported 97% precision with 72% recall (F-score = 0.83) while ARGH (Wren and Garner, 2002) reported 96% precision with 93% recall (F-score = 0.94). The results (abbreviations and their full form) of Stanford University's Abbreviation Server, SaRAD, AcroMed and ARGH were available (Wren et al., 2005). Acromine (Okazaki and Ananiadou, 2006) used the observation that abbreviations can usually be expanded to their full form (reversing the idea of reducing full forms into abbreviations) reported precision of 99% with 82% to 95% recall (F-score = 0.89 – 0.97). Sohn et al. (2008) uses a similar long-form to abbreviation matching algorithm to AbbRE (Yu et al., 2002b) reported 96.5% precision with 83.2% recall. MBA (Xu et al., 2009) used text alignment for acronym-type abbreviations and statistics for non-acronym-type abbreviations. It reported 88% recall and 91% precision (F-score = 0.89). Torii et al. (2007) performed a meta-study to compare the results of a number of AR systems and found that they generally agree with each other in terms of results.

### **1.5.5. Named Entity Recognition: Identifying the Players**

The goal of Named Entity Recognition (NER) is to find entities, the names of physical or abstract objects, in a given text (Franzen et al., 2002); in particular, the names of chemicals, proteins and genes. Essentially, NER asks the question “What makes a name a name?” This question is generally answered by recognizing words that may refer to entities, followed by identifying the entities in question uniquely. NER is currently one of the most difficult tasks in biomedical text mining (Jensen et al., 2006) and solving this problem will allow for more complex text mining tasks to be addressed (Hanisch et al., 2003) as it is a prerequisite for information extraction and advanced IR (Jensen et

al., 2006; Krauthammer and Nenadic, 2004; Proux et al., 1998). NER could also be expanded into recognizing other medically important terms, such as names of diseases (Jimeno et al., 2008).

One of the main reasons for the difficulty is the high degree of variations in terms that are not explicitly reflected in biomedical ontologies (Nenadic et al., 2004). It is common that biological entities can have several names, for example, PTEN and MMAC1 refers to the same entity (Cohen and Hersh, 2005). It was estimated that one-third of biological terms are variants (Jacquemn, 2001). In addition, biological and chemical entities may have multi-word names and variants of the names. For example, “Peroxisome Proliferator Activated Receptor”, “Peroxisome proliferator-activated receptor” and “Peroxisome-proliferator-activated receptor” refer to the same entity. Liu et al. (2006) did a comprehensive study on this area. With new genes and proteins being discovered and named in the genomic era, it can be implied that there is no complete dictionary of named biological entities, hence, NER by simple text matching will not suffice (Cohen and Hersh, 2005; Tsuruoka and Tsujii, 2004). Because of the potential utility and complexity of the problem, NER has held the attention of many researchers, and it is not surprising that much work in biomedical NER had focused on recognizing gene and protein names in text.

The approaches can be classified as either lexicon-based, rules-based, statistics-based, or combinations of these means. Krauthammer et al. (2000) adapted BLAST algorithm to identify entities from text with 78.8% precision and 71.7% recall (F-score = 0.75). A good example of a rules-based NER system is AbGene, which was based on a Brill tagger trained on 7000 manually-tagged sentences using a 'Gene' tag. It achieved a 85.7% precision and 66.7% recall (F-score = 0.75) (Tanabe and Wilbur, 2002). In contrast, GAPSCORE, also a rules-based system, examines the appearance, morphology and context of the word before applying a classifier trained using these features. It achieved 74% precision and 81% recall (F-score = 0.77) in inexact matches, and 59% precision and 50% recall (F-score = 0.54) in exact matches (Chang et al., 2004). On the other hand, Hanisch et al. (2003) used a dictionary approach and achieved 95% precision and 90% recall (F-score = 0.92) while Egorov et al. (2004) achieved 98% precision and 88% recall (F-score = 0.93). A further study report using curated

dictionary demonstrated good performance, F-score between 0.8 to 0.9, across different organisms (Hanisch et al., 2005). VTag (Ryan et al., 2004), McDonald and Pereira (2005) and ABNER (Settles, 2005) employed conditional random field, a statistical approach and achieved precisions between 58.2% to 85.4% and recall between 53.9% and 79.8%. Zhou et al. (2005) combined several approaches using voting strategy to achieve a F-score of 0.83. Other groups had also attempted combinations of approaches to improve precision (Hatzivassiloglou et al., 2001; Hou and Chen, 2004; Majoros et al., 2003; Finkel et al., 2005). Jimeno et al. (2008) demonstrated that although a curated dictionary method for NER is still superior (F-score of 0.59 to 0.69) compared to statistical methods (F-score of 0.28 to 0.57), voting strategy may provide a better performance than any single method (F-score of 0.54 to 0.83).

It is clear from recent developments that a dictionary approach (solely or in combination) tends to outperform lexicon-based or statistics-based approaches (Kou et al., 2005; Fundel et al., 2005; Jimeno et al., 2008). However, it is debatable how well NER must perform before rendered useful in biomedical text mining (de Bruijn and Martin, 2002) as previous studies illustrated that performance (in terms of F-score) of biomedical information extraction is approximately equal to that of biological NER (Gaizauskas et al., 2003; Daniel et al., 2004; Wren and Garner, 2004).

It is unlikely automated biomedical NER will approach that of human experts in the near future in terms of precision. However, current biomedical NER systems can be useful in providing an initial list of genes and protein names for human curation if precision is critical in the context of application.

A close relative to NER is an area of study known as gene normalization (GN). The main purpose of GN is to standardize a set of gene names using a thesaurus of gene synonymes and homologies (Crim et al., 2005) which will be useful in many aspects of biomedical literature analysis. GN is currently an area of active research and had warranted a specialized track in the latest BioCreative II challenge workshop in 2007 (Hakenberg et al., 2008; Huang et al., 2008; Krallinger et al., 2008a).

Intuitively, AR is an easier task than NER, which is supported by the observation that

current AR systems uniformly perform consistently better compared to current NER system in term of their F-score (Cohen and Hersh, 2005). Similarly to NER, AR techniques have been used to create lists of abbreviations, such as ADAM (Zhou et al., 2006b), which can be used for other applications.

### **1.5.6. Information Extraction: Getting the Facts**

The most common aim of biomedical information extraction (IE) is finding relationships between two entities (Hoffmann et al., 2005; Skusa et al., 2005), in this case, usually either genes, proteins or metabolites. The relationship in question may range from very general, like any form of biochemical association, to very specific, such as regulatory activation. In contrast to IR, IE systems tends to be less *ad hoc* but are more targeted towards specific relationships. Another difference is the granularity of text. IR systems identifies text of interest whereas IE systems work within the text to identify facts of interest, which can be subsequently verified by a curator reading the paper of interest. Biomedical information extraction are currently being developed by three different ways (Cohen and Hunter, 2008); co-occurrence, template matching, and natural language processing.

#### **1.5.6.1. Co-occurrence**

Co-occurrence is fundamentally statistical and based on the tenet that multiple occurrences of the same pair of entities suggests that the pair of entities are related in some way (Stapley and Benoit, 2000; Wilbur and Yang, 1996; Cohen and Hunter, 2007) and the confidence of such relatedness increases with more co-occurrences. In practice, most systems used a frequency-based scoring technique (Donaldson et al., 2003; Stapley and Benoit, 2000; Hoffmann and Valencia, 2004; Jensen et al., 2006) to ensure that the co-occurrence of two entities is higher than random chance (Cohen and Hersh, 2005). Being a statistical probability, co-occurrence of entities within the same text alone will not give any insights into the type and nature of the relationship (Stephens et al., 2001) unless it is used downstream to IR systems which pre-identified the type of relationships of interest (Donaldson et al., 2003; Cooper and Kershenbaum, 2005; Ray and Craven, 2005). Despite so, the advantages of co-occurrence techniques over natural language processing are simplicity, easy to implement and efficient over large amounts

of data (Jelier et al., 2005).

Two of the most successful implementations of co-occurrence methods are PubGene (Jenssen et al., 2001) and CoPub Mapper (Alako et al., 2005). Both had been shown to co-relate well with microarray results.

A variant of co-occurrence which bootstrapped on PubMed had also been suggested (Becker et al., 2003; Simon et al., 2004) and is commonly known as co-citation. Common experiences in using PubMed suggested that if two terms were used together with an 'AND' clause to search PubMed, it would return the intersection of the results compared to when each term was used separately. Translated statistically, the size of the intersection (proportion of co-cited documents) increases with more relatedness in the pair of entities used to interrogate PubMed. However, co-citation had not been evaluated against co-occurrence measures.

#### **1.5.6.1.1. Brief Descriptions of Co-Occurrence Systems**

PubGene (Jenssen et al., 2001) used the idea that if 2 entities were mentioned in the same article, there will be some relationship, no matter how remote. By this, it simply counted the number of articles with occurrences of 2 entities in question and used the count as a relative strength of relatedness. If there was 1 article in 10 million mentioning both gene entities, there would be 60% chance of a true relationship between them. This was increased to 71% with 5 or more articles in 10 million.

CoPub Mapper (Alako et al., 2005) calculated the article number-normalized occurrence of any 2 entities (number of articles with both concepts divided by total number of articles, divided by the product of number of articles with one concept each divided by the total number of articles), which was termed as mutual information measure. Mutual information measure was then converted to logarithmic scale and normalized on a scale of 0 to 100, known as the scaled log transformed relative score. The confidence of relationship between 2 entities was directly proportional to the scaled log transformed relative score.

MedInfoText (Fang et al., 2008) aims to extract the relationships between gene



methylation and cancer from biomedical literature. It uses Lucene-based full text search engine for Perl, Plucene (<http://search.cpan.org/dist/Plucene/>), for term indexing. The relationships are extracted based on co-occurrences of terms in abstracts and sentences by measuring association rule interestingness (Han and Kamber, 2006) using support and confidence measures.

#### **1.5.6.2. Natural Language Processing / Template Matching**

There is significant overlap between Natural Language Processing (NLP) methods and Template Matching methods as both share many common components, such as ontologies and substantial use of Regular Expressions. Template Matching methods mainly use the grammatical structure of English sentence to construct templates, which can be done either manually (Yu et al., 2002a) or automated (Huang et al., 2004; Yu and Agichtein, 2003), and using these templates to extract specific information from text. An example has shown extracting mutation information solely by regular expressions achieved 85% precision (Florence et al., 2004). BioIE (Divoli and Attwood, 2005) uses rules with template matching. However, sole use of template matching is not widespread in biomedical text analysis. Instead, template matching is usually used either implicitly within NLP methods or explicitly to process the outputs of NLP. The main principles of NLP will be described to facilitate discussions into the tools that employed NLP.

Natural Language Processing (NLP) covers all aspects and stages of processing natural human speech or text into forms usable by automated systems. In the case of biomedical NLP, it can be reasonably assumed that only machine-accessible text in English language forms the majority of source materials. Hence, this section only covers the processing of English text. Given the long history of NLP, a large volume of text had been written and this section can only provide a broad coverage of the general techniques used in NLP, namely, tokenization, part of speech (POS) tagging, and shallow parsing.

The text is first broken up into its constituent atoms, known as tokens, by a process of tokenization. While the granularity of tokens may vary from chapters to phonemes (atomic units of sound), the most common form of tokenization for NLP is to break

down each sentence into words and punctuations. Generally, in English text, words in a sentence are delimited by whitespace(s), except words prior to a punctuation, like a comma. This feature poses the main challenge of tokenization – distinguishing punctuations (especially period) signaling the end of either a sentence or phrase and that being part of the previous token, like in shorthand (for example, Mr., Dr.). Another challenge is the expansion of common contractions, such as “you’ve”, which is usually done using a dictionary approach.

Part of Speech (POS) Tagging is the process of annotating a series of tokens, presumably a sentence, with semantics information (that is, the role of each token in a sentence) with a set of tags, such as Penn Treebank Tag Set (Marcus et al., 1993). There are two main approaches to POS tagging, rule-based and probabilistic. Probabilistic taggers estimate the probability of a sequence of POS tags for a given sequence of words, based on a probability model, such as the Hidden Markov Model (Dernatas and Kokkinakis, 1995; Kupiec, 1992). On the other hand, a rule-based tagger (Brill, 1995) uses contextual rules to assign tags to ambiguous words. Contextual rules, often known as context frame rules, are rules suggesting a tag based on the tag(s) before and after the unknown word. This is usually followed by morphological rules which looks at the appearance of the word. For example, words ending with “ing” is likely to be verbs.

POS Tagging can be seen as a reduction scheme to map a potentially infinite amount of words into a small and definite set of tags, to facilitate further processing. Based on the sequence of POS tags, the source text is then broken up into non-overlapping phrases. This process is chunking, also known as shallow parsing, where phrases are tagged by a small number of grammatical phrase tags, such as, *Noun Phrase*, *Verb Phrase*, *Prepositional Phrase*, *Adverb Phrase*, *Subordinate Phrase*, *Adjective Phrase*, *Conjunction Phrase*, and *List Marker*. Chunking is generally useful as a preprocessing step for information extraction as the main effect of NLP is to process unstructured data (in the form of human text) into a more structured form (POS annotated phrases), suitable for information extraction (Saric et al., 2005; Friedman et al., 2001; Temkin and Gilder, 2003; Rzhetsky et al., 2004) or phrase extraction in itself, can be used for medical concept identification (Li and Wu, 2006). One of the most important output of chunking is the subject-verb-object(s) tuples. In linguistic typology, the English

language, together with more than 75% of all languages in the world is classified under the subject-verb-object (SVO), also known as the agent-verb-object (AVO) typological system (Crystal, 1997). A sentence can be processed into one more more SVO tuples, which are useful for extracting relationships between entities (Santos et al., 2005; Uramoto et al., 2004; Rindflesch et al., 2000). In contrast to shallow parsing (chunking), full parsing or complete parsing is much more computationally intensive but has been shown to be useful in a real world biological application (Rinaldi et al., 2007).

#### **1.5.6.2.1. Brief Descriptions of Natural Language Processing / Template Matching Systems**

GENIES (Friedman et al., 2001) used GenBank and SwissProt to tag genes and protein names in text before processing using MedLEE (Friedman, 2000), a specialized text processor for biomedical literature, which used rules to parse text into structured frames. The original lexicon in MedLEE was enlarged in GENIES.

MedScan (Novichkova et al., 2003) used a biomedical lexicon to tag text before tokenization and stemming words into their infinite form. Tokens were syntactically processed by a parser based on active chart parser algorithm (Allen, 1994). Syntactic tokens were processed into semantics structured based on an established method (Sells, 1984).

MeKE (Chiang and Yu, 2003) used Gene Ontology and LocusLink as basis for constructing function names and gene names ontologies to tag text. A pattern recognizer was trained to recognize sentences or phrase describing gene functions by sentence alignment. Extracted sentences were classified by the probability of containing protein-function relationships by a Naïve Bayes classifier.

PreBIND (Donaldson et al., 2003) collected a list of non-redundant protein names from the NCBI RefSeq database, which was used to scan for protein names in text. Extracted term features from text (protein names, words, adjacent words) were used to train a linear support vector machine for identifying protein-protein binding interactions using yeast data.

Arizona Relation Parser (Daniel et al., 2004) specialized a Brill tagger (Brill, 1995) with 100 PubMed abstracts and GENIA corpus (Kim et al., 2003). POS tagged text was processed by a hybrid shallow parser which allowed for multilevel n-ary branching of up to 24-nary, meaning POS tags or phrases up to 24 tags or phrases away could be linked. Information from each allowed combination was extracted by rule templates.

BioRAT (David et al., 2004) used GATE (Cunningham, 2000) to provide utilities to perform POS tagging and information extraction. Text was POS tagged to remove uninformative words, such as determinant verbs, before passing through a series of template matching to extract protein-protein interactions using a set of handwritten templates and a manually curated list of entities forming the gazetteers. An evaluation between information extraction from abstracts and full text demonstrated that 58.7% of the interactions were extracted from full text but the precision from full text was 51.3%, 3.82% lower than that of abstracts (55.07%).

Chilibot (Chen and Sharp, 2004) used TnT POS tagger (Brants, 2000) trained on GENIA corpus (Kim et al., 2003) followed by CASS for shallow parsing. Chunked text was used to construct named interactions among either biological concepts, genes, proteins, or drugs. It also showed that the connectivity of molecular networks extracted from the biological literature follows the power-law distribution.

GIS (Chiang et al., 2004) consists of two modules: gene information screening and gene-gene relation extraction. In gene information screening, documents were downloaded from PubMed and informative sentences were selected before tagged using a domain-specific lexicon built from online dictionaries and suggestions by biomedical researchers. Gene-gene relation extraction had a identifier for gene-gene relations, presumably trained by machine learning methods. Identified relations were then evaluated to be positive, cooperative or negative.

MedTAKMI (Uramoto et al., 2004) was built on TAKMI, which used standard text processing (tokenization, POS tagging, shallow parsing) to process text into (Nasukawa and Nagono, 2001) subject-verb-object(s) tuples as an intermediate form for further information extraction. Besides using a dictionary of protein names, the difference

between TAKMI and MedTAKMI was not clear. A number of tools for information extraction from subject-verb-object(s) tuples has been described (Uramoto et al., 2004) but none was evaluated. Nevertheless, MedTAKMI could be used as a curator's tool.

Karopka et al. (2004) used GATE (Cunningham, 2000) and modified the ANNIE gazetteer of GATE to tag for gene names before tokenizing the sentences and POS tagging by GATE. Gene relations were extracted by 34 manually-written grammar rules in JAPE (Java Annotation Patterns Engine) language, which is essentially template matching.

Cooper and Kershenbaum (2005) overlapped the results of text processing, and graphical and statistical analysis to extract protein-protein interactions. TALENT text mining system (Neff et al., 2004) was used for extracting protein-protein interactions from text by a method previously established for extracting relationships between noun phrases (Cooper and Byrd, 1998). For each pair of proteins, a 3-hop neighbourhood graph was defined and the coherence of the graph, defined as “*the ratio of the number of edges present to the possible number of edges*”, was calculated but the threshold coherence for a positive result was not obvious. Positive results from both methods, led to an improvement of precision from 62% to 74%.

Santos et al. (2005) used a shallow parser, CASS, to extract protein names (Wnt pathway proteins) from text by statistical comparison with a Wnt pathway corpus to avoid maintenance of a list of protein names. At the same time, they used Link parser to process text to subject-verb-object(s) tuples before full parsing to extract interactions between the Wnt pathway components. However it was not clear which POS tagger was used before shallow parsing by CASS nor which grammar was used for Link parser.

Jang et al. (2006) avoided the problem of multi-word protein names and complex sentences by simplification of sentences – protein names and specific noun phrases were substituted by pre-defined words, and parenthesis phrases which does not contain entity names were removed. This simplified sentence is POS tagged by a Brill tagger (Brill, 1995) trained on GENIA corpus (Kim et al., 2003), then shallow parsed. Protein-protein interactions were extracted by Regular Expression parsing of shallow parsed sentences.

CONAN (Malik et al., 2006) combined several known tools and used a set of decision criteria to evaluate the output of these tools and achieved 53% precision and 52% recall using LLL corpus (Cussens and Nedellec, 2005). CONAN used Krauthammer et al. (2000), AbGene (Tanabe and Wilbur, 2002) and NLProt (Mika and Rost, 2004) for NER, and MuText (Horn et al., 2004) and PreBind (Donaldson et al., 2003) for interaction extraction.

GeneLibrarian (Chiang et al., 2006) is a PubMed text summarization tool that uses a list of gene names as input, instead of keywords. It consists of 2 modules: GeneCluster and GeneSum. GeneCluster clusters the list of given gene names using Gene Ontology while GeneSum obtains text from PubMed based on the clusters and process the text linguistically by natural language processing methods. Finally, a 9-state finite state machine (FSA) is used to perform text summarization based on the part-of-speech tags of the processed text.

Rinaldi et al. (2007) used a dependency parser, Pro3Gres (Schneider et al., 2004), to parse processed text. It reported precision of 96% and recall of 63%. The source text were initially analyzed for specific terms, such as entity names, before splitting into sentences by MXTERMINATOR (Reynar and Ratnaparkhi, 1997) and tokenized using Penn Treebank tokenizer (Marcus et al., 1993). The tokenized text was POS tagged by MXPOST (Ratnaparkhi, 1996) and lemmatized by morpha (Minnen et al., 2001). After which, the text was processed against GENIA Ontology (Kim et al., 2003) before shallow parsed by LTCHUNK (Mikheev, 1997). The chunks were parsed for dependencies by Pro3Gres.

Feng et al. (2007) aimed to extract chemical-CYP3A4 interactions from biomedical text. They had used a combined rule-based and dictionary-based method to identify chemical names in text and had used GATE (Cunningham, 2000) for POS tagging and information extraction. An evaluation with 100 abstracts demonstrated 87.4% recall and 92.3% precision for chemical name identification and 85.2% recall and 92.0% precision for the extraction of chemical-CYP3A4 interactions.

E3Miner (Lee et al., 2008) aimed to extract the interactions between ubiquitin-protein ligase (E3) and its target proteins from biomedical text. E3 was identified using a specially constructed POS tagger and shallow parser. The target proteins were then identified using a rule-based method before processing for Gene Ontological terms. Using a set of 47 abstracts, a precision of 97% and a recall of 74% was indicated.

PIE (Kim et al., 2008) used a 2-phase method: a term co-occurrence based method to identify potential abstracts that may contain protein-protein interactions, followed by NLP processing using a POS tagger trained on GENIA corpus (Kim et al., 2003) to extract protein-protein interactions. PIE was tested on BioCreAtIvE I corpus (Plake et al., 2005) and reported 84% precision.

#### **1.5.6.3. Applications of Biomedical Information Extraction**

There are two main schools of thought in current biomedical IE, one school (call it the “Specialist” for further argument) takes the view that biomedical texts are specialized text (very much like the use of *Legalese* in legal documents) requiring highly domain-specific tools. This opinion had sparked off the development of biomedical-specific POS tag sets (such as SPECIALIST tag set (National Library of Medicine, 2003)), POS taggers (such as MedPost (Smith et al., 2004)), ontologies and NLP systems (such as MedLEE (Friedman et al., 1994)). Another school (call it the “Generalist”) takes the view that biomedical texts are not sufficiently specialized to require a re-development of existing tools but either re-use or adapt generic NLP tools for biomedical text processing. This triggered the use of generic NLP systems, such as TAKMI (Nasukawa and Nagono, 2001), Link Grammar (Sleator and Temperley, 1991), and GATE (Cunningham, 2000), for biomedical IE.

Regardless of opinions, the focus of biomedical IE has been on a few types of relationships, namely, physical protein-protein interactions (PPIs) (Donaldson et al., 2003; Cooper and Kershenbaum, 2005; Friedman et al., 2001; Hao et al., 2005), non-physical PPIs (von Mering et al., 2005; Daraselia et al., 2004; Yakushiji et al., 2001), relationships between proteins and diseases and terms (Alako et al., 2005; Chen et al., 2008; Osborne et al., 2007; Tiffin et al., 2005), gene regulation (Aerts et al., 2008; Saric et al., 2005), protein phosphorylation (Hu et al., 2005b; Lee et al., 2008), alternate

transcription (Shah et al., 2005), and functions of transcription factors (Yang et al., 2008b).

Most of the earlier work in biomedical IE belongs to the Specialist's School and one of the significant contributions was the GENIES system (Friedman et al., 2001), which modified MedLEE's lexicon preprocessor and parser. GENIES was only evaluated using one article and reported an overall precision of 96%. MedLEE (Friedman, 2000) was also adapted to process pathology reports for breast cancer study (Xu et al., 2004a). The MeKE system used a lexicon of gene and protein names from LocusLink to construct an ontology for pattern matching within text into structured data (Chiang and Yu, 2003). Novichkova et al. (2003) agreed that NLP can be used to process text into semantic structures and developed MedScan, which incorporated a biomedical lexicon. Further work by Daraselia demonstrated 91% precision with 21% recall (F-score = 0.34) in extracting protein interactions from text using MedScan (Daraselia et al., 2004). The output of MedScan was assembled and constructed into a set of tissue-specific pathways, ResNetCore database (Yuryev et al., 2006). The Arizona Relation Parser (Daniel et al., 2004) attempted to improve MedScan's low recall by re-training Brill Tagger (Brill, 1995) with Brown Corpus, Wall Street Journal, PubMed abstracts, and added GENIA lexicon (Kim et al., 2003). This was followed by a hybrid grammar, template matching and semantic filtering. It reported 89% precision with 35% recall (F-score = 0.5). GIS (Chiang et al., 2004) uses a domain-specific lexicon but instead of NLP, it employs a machine learning approach and reported 84% precision with 77% recall (F-score = 0.80). Jang et al. (2006) had trained Brill tagger (Brill, 1995) on GENIA corpus (Kim et al., 2003) and a purpose-built protein-protein interaction extractor system which achieved 81% precision and 43% recall (F-score = 0.56). E3Miner built a specialized POS tagger and shallow parser to achieve 97% precision and 74% recall (F-score = 0.84). PIE (Kim et al., 2008) used a GENIA (Kim et al., 2003) trained POS tagger and achieved 84% precision.

In the Generalist's School, BioRAT (David et al., 2004) is one of the earliest systems to modify GATE (Cunningham, 2000) to extract protein-protein interactions and reported 48% precision with 39% recall (F-score = 0.43). TAKMI (Nasukawa and Nagono, 2001), originally developed to process customers call logs in IT helpdesks, was used to



develop MedTAKMI (Uramoto et al., 2004) which uses term frequency to support search results. Karopka et al. (2004) modified the ANNIE system of GATE (Cunningham, 2000) and modified precision and recall measures to account for partial correct extractions, and reported 92.8% precision with 30% recall. Cooper and Kershenbaum (2005) used a generic TALENT text mining system (Neff et al., 2004) with graphical and statistical approaches to mine for protein-protein interactions and reported 74% precision. A Link grammar parser (Sleator and Temperley, 1991) was used to mine the Wnt pathway and reported 90% precision with 64% recall (F-score = 0.75) (Santos et al., 2005). Rinaldi et al. (2007) used a myraid of text processing tools with GENIA Ontology (Kim et al., 2003) and reported 96% precision with 63% recall (F-score = 0.76). Feng et al. (2007) used a purpose-built rule-dictionary hybrid for chemical name identification and GATE (Cunningham, 2000) for information extraction and reported 85% recall and 92% precision (F-score = 0.87).

Table 1a. Summary of performances of biomedical literature analysis systems. 'NG' means that the particular performance metric was not given in the study.

Specialist Systems			Generalist Systems		
Study	Precision	Recall	Study	Precision	Recall
Friedman et al., 2001	0.96	NG	David et al., 2004	0.48	0.39
Daraselia et al., 2003	0.91	0.21	Karopka et al., 2004	0.93	0.30
Daniel et al., 2004	0.89	0.35	Cooper and Kershenbaum, 2005	0.74	NG
Chiang et al., 2004	0.84	0.77	Santos et al., 2005	0.90	0.64
Jang et al., 2006	0.81	0.43	Rinaldi et al., 2007	0.96	0.63
Lee et al., 2008	0.97	0.74	Feng et al., 2007	0.92	0.85
Kim et al., 2008	0.84	NG			

From the research results gathered from both schools of thought (tabulated in Table 1a), it is still not possible to demonstrate superiority of one approach over the other in terms of system performance. Intuitively, it might be easier to modify an existing system for a specific application than to develop one from scratch. In addition, it has not been demonstrated that systems developed from the Specialist's school of thought can be adapted to extract other biomedical relationship of interests as only a few systems have been designed to extract multiple relationships (Jensen et al., 2006). On the other hand,

Generalists view generic NLP systems as a processing tool to convert unstructured text into structured forms, such as tuples; thus, is inherently more readily adapted for different problems. It is probably reasonable to comment that by adapting an existing system for use in biomedical text mining usually implies that the system has been used in a different context. For example, TAKMI was used in 3 different areas; analyzing customer call logs (Nasukawa and Nagono, 2001), generating frequently-asked-questions candidates (Matsuzawa and Fuduka, 2000), and in MedTAKMI (Uramoto et al., 2004).

However, adapting a generic system may require intense effort in formulating rules and templates (GATE and Link Grammar) for the specific problem domain or re-training parts of the system, especially POS tagger, which may require a prior manual tagging of training corpus (Jensen et al., 2006). For instance, Chilobot (Chen and Sharp, 2004) used TnT tagger (Brants, 2000) trained on the GENIA corpus (Kim et al., 2003) but succeeded in using CASS parser (<http://www.vinartus.net/spa>), un-modified, for chunking. This might be an obstacle to adapt a previously adapted generic NLP system for a biomedical problem to another biomedical problem. Moreover, there is no certainty of rewards in this effort as Miyao et al. (2009) had shown that combining text processing components may be synergistic.

It is inherent in the process of evaluating systems using corpora that human experts are the only absolute performer. That is, human experts are performing at 100% precision and 100% recall. Therefore, it should be conceivable that learning from the output errors by artificial intelligence and machine learning methods could be used to improve information extraction systems. The first of such biomedical information extraction systems which uses support vector machines and neural networks to mimic human expert curation had surfaced (Rodriguez-Esteban et al., 2006) with precision ranging from less than 30% to more than 90% over 68 extraction tasks. In addition, biomedical information extraction had been shown to be able to improve curation efficiency of protein-protein interactions into database (Alex et al., 2008).

### **1.5.7. Text Mining: Finding Hypotheses**

While the main premise of IR and IE is deductive reasoning (the conclusion is of no

greater generality than the premises), text mining (TM) is fundamentally inductive reasoning (the conclusion is of greater generality than the premises). In other words, TM aims at finding or induce new information and hypotheses from existing knowledge from the literature. One of the pioneers of biomedical TM is Don Swanson who suggested in the mid-80s that there were connections between fish oil and Raynaud's syndrome (Swanson, 1986), migraine and magnesium (Swanson, 1988), arginine intake and the level of somatomedin C in blood (Swanson, 1990b). This had triggered the advancement of biomedical IR/IE, NER and AR, which are all precursors to TM. The method Swanson used is essentially *Hypothetical Syllogism* (if p then q; if q then r; therefore, if p then r), which is an extension of *Modus Ponens*. In biomedical TM, it is commonly referred to as Swanson's ABC model (Swanson, 1990a). Using this discovery model, Weeber et al. (2005) had attempted to automate it and found new potential uses for thalidomide. More recently, the potential therapeutic use of turmeric on spinal cord injuries was suggested (Srinivasan et al., 2004).

Despite its potential and history, biomedical TM is still at its infancy (Bekhuis, 2006). In order for hypothesis generation systems to be a standard tool of biologists, a fundamental question needs addressing – how to evaluate an untested set of hypotheses? (Cohen and Hersh, 2005) A way to circumvent this problem may be using statistical measurements from IR/IE to provide a means of prioritizing the potential of each hypotheses, as shown as Anne 2 (Jelier et al., 2008). In spite of this inherent problem, there may be use of TM to evaluate and score several possible hypotheses from experimental or clinical research (Smalheiser et al., 2009). TM is also known by other authors as “literature based discovery” (Weeber et al., 2005, Sarkar and Agrawal, 2006, Hristovski et al., 2005) or “knowledge discovery” (Yetisgen-Yildiz and Pratt, 2006).

### **1.5.8. Related Areas of Importance**

Notwithstanding the development in previously discussed areas, there are three other key areas that are important within the literature analysis pipeline, namely, corpora, which forms the gold standard for evaluating systems; databases, which may be used to evaluate systems or are themselves resulting from literature analysis systems; evaluation strategies, the definition of performance metrics and their calculations; assisted microarray analysis using output from biomedical text analyses; and visualization tools

for viewing large interaction maps.

#### **1.5.8.1. Corpora**

A corpus (corpora for plural) is a collection of literature which has been either tagged, annotated or categorized for specific purpose(s). Essentially, a corpus is a defined data set of literature. The importance of corpora to literature analysis tools cannot be over-emphasized. Analogously, it is as important as antibodies to protein studies. However, there are not many corpora of biomedical literature for various purposes as they often require manual annotations with high-level agreement among annotators (Colosimo et al., 2005; Wilbur et al., 2006), known to be labour-intensive to create (Jensen et al., 2006) and need to reflect a biologist's interpretation of the text (Kim et al., 2008). The main value of corpora is that it provides a known finite source of positives which is essential for calculating recall measure and error analyses.

Categorically, the following biomedical corpus for different purposes are as follows: For protein and gene name recognition (NER), there are Yapex (used in (Chang et al., 2004)) and GeneTag (Tanabe et al., 2005), PennBioIE (Ryan et al., 2004) corpora. For abbreviation recognition, there is Medstract (Pustejovsky et al., 2002) corpus. For part-of-speech tagging, there are GENIA (Collier et al., 1999; Kim et al., 2003; Ohta et al., 2002), PennBioIE (Ryan et al., 2004 ) and MedPost (Smith et al., 2004). GENIA team had also expanded the annotation into biomedical events to reflect a biologist's understanding of the text (Kim et al., 2008). For relationship extraction, there is a dataset used for Learning Logic in Language 2005 ([www.cs.york.ac.uk/aig/lll/](http://www.cs.york.ac.uk/aig/lll/)) (Cussens and Dzeroski, 2000; Cussens and Nedellec, 2005) and BioCreAtIvE corpus (Plake et al., 2005). BioScope corpus (Vincze et al., 2008) represents the first attempt to incorporate uncertainty or negative information into a corpus.

There is a general sentiment that progress in biomedical literature analysis suffers from the lack of corpora (Leser and Hakenberg, 2005) which is relatively obvious when one starts to list down the corpora available for each purpose. There is no biomedical corpus for shallow parsing (chunking) or citation retrieval from PubMed (information retrieval) and minimal choices for relationship extraction. Moreover, testing using different corpora (if any) can result in F-score varying as much as 19% (Pyysalo et al., 2008).

Hence, researchers had resorted to compare their system output with that in curated databases (Zhang et al., 2005) as these databases represent high-quality molecular interaction data (Chatr-aryamontri et al., 2008).

#### **1.5.8.2. Databases**

The main database for biomedical literature is PubMed where most source materials for literature analysis work is derived from. Possibly, the largest repository for biochemical information is KEGG which provides links to GenBank and a number of other publically available databases. Databases are repositories of source text (PubMed), curated tools for comparison (comparing system output against KEGG in Zhang et al., (2005), Lee et al. (2007) compared their system against SwissProt and Maguitman et al. (2006) tested their system against Pfam), or are themselves the results from literature analysis, such as DIP (Salwinski et al., 2004). It is generally true that databases can benefit from literature analysis efforts (Miotto et al., 2005). Large institutional initiatives, such as KEGG and BIND (Alfarano et al., 2005), which are mainly manually maintained and curated, cater to the general research community.

*Bioinformatics* has a section of the periodical catering to the publications of databases and *Nucleic Acid Research* releases issues periodically with a database focus (known as *database issue*). The latest edition of *Nucleic Acid Research database issue* (Volume 36) features 98 databases. These have almost become a *de facto* source of new databases. As the availability of corpora is scarce (Zhou and He, 2008), a cursory knowledge of database availability might assist in evaluation efforts.

In terms of individual proteins, there are databases for proteins in specific organelles (Scott et al., 2004; Basu et al., 2006); prokaryotic proteins (Martinez-Bueno et al., 2004); proteins of specific biochemical events (Holliday et al., 2005; Mao et al., 2005); proteins of specific domains or characteristics (Magkrioti et al., 2004; George et al., 2004; Li and Gallin, 2004); protein anomalies (Vucetic et al., 2005); transcription factors (Guo et al., 2005; Gao et al., 2006); and specific classes of proteins, such as lectin (Chandra et al., 2006). Some databases focused on protein-protein interactions, which may be all types of interactions (Salwinski et al., 2004; Lin et al., 2005; Goll et al., 2008; Pawlicki et al., 2008; Theodoropoulou et al., 2008) or specific interactions

(Yimeng et al., 2004; Beuming et al., 2005; Dou et al., 2004; Yang et al., 2008a).

For genes, there are databases for genes of specific characteristics (Mao et al., 2005; Bobby et al., 2005; Sakharkar et al., 2004); cleavage sites (Brockman et al., 2005); promoters or regulatory elements (Palaniswamy et al., 2005; Kim et al., 2005; Gallo et al., 2006; Morris et al., 2008; Rushton et al., 2008); genes of specific organisms (Liu and Yang, 2004; Kaas et al., 2008); organelle genomes (Basu et al., 2006); entire genomic resource (Gauthier et al., 2007); and expressed sequence tags (Beldade et al., 2006).

Other databases include those for managing experimental results (Sherlock et al., 2001; Markus et al., 2004; Li et al., 2005); haptens (Singh et al., 2006); orientation of proteins on cellular membranes (Lomize et al., 2006); protein localization (Zhang et al., 2008); and therapeutically important pathways (Zheng et al., 2004). Currently, the largest and most extensive database for microarray and other high-throughput data storage is the Gene Expression Omnibus (GEO) (Barrett and Edgar, 2006).

Scanning the wide variety of databases, it is clear that the challenge is not in the creation but on the use of these databases, especially integrating them into a composite (federation) of biomedical databases and querying them (Lakshmanan et al., 2001; Wyss and Robertson, 2005; Jeffery et al., 2007; Fristensky, 2007; Liu et al., 2006b; Garcia Castro et al., 2005).

#### **1.5.8.3. Evaluation Strategies**

During the development of a literature analysis tool, it is critical to have an estimation of the reliability, which are usually compared to a standard of desired results (Cohen and Hunter, 2008), usually in a form of tagged, annotated or categorized corpus. The most common evaluation strategy and metrics (measurements), such as precision and recall, originated from the Second Message Understanding Conference in 1989.

Given a corpus and a specific query, the results can be partitioned into true positives (TP; items correctly labeled as positive), false positives (FP; items incorrectly labeled as positive), true negatives (TN; items correctly labeled as negative), and false negative

(FN; items incorrectly labeled as negative). With these four items, a few metrics can be established. the most common being precision, also known as positive predictive value, is defined as  $TP/(TP+FP)$ ; recall is  $TP/(TP+FN)$ . Precision and recall are typically inversely related (Zhou et al., 2006).

Precision and recall are commonly used because of their simplicity to evaluate against an established standard (annotated corpus). However, in the absence of a standard, precision can still be evaluated comparing the output of a system with its input. Karopka et al. (2004) modified precision and recall measures to account for partially correct extractions. Precision and recall are important because the inverse of precision is a measure of false positives ( $1 - \text{precision}$ ) of the system output and the inverse of recall measures false negatives ( $1 - \text{recall}$ ) or proportion of lost information as a result of processing.

It is important to note that in IE, it is not possible to define true negatives (TN) as there are no theoretical bounds of the number of 'facts' that can be generated from a piece of text (Hirschman et al., 2002). Therefore, a number of measures that required TN, such as accuracy  $((TP+TN)/(TP+FP+TN+FN))$ , error rate  $((FP+FN)/(TP+FP+TN+FN))$ ; which is  $1 - \text{accuracy}$ , negative predictive value  $(TN/(FN+TN))$ , prevalence  $((TP+FN)/(TP+FP+TN+FN))$ , and specificity  $(TN/(TN+FP))$  are impossible to calculate. In addition, receiver operating characteristics (ROC), which had been used extensively in evaluating system performance (Biagini et al., 2001; Gjengsto et al., 2005; Margolis et al., 2002; Rosman and Korsten, 2007), cannot be calculated for IE as it requires specificity.

Notwithstanding variations using different corpora for evaluating different systems and using different criteria for assigning results into each of the three bins (TP, FP, FN), it will be difficult to compare two systems each characterized by precision and recall. Given presence of a single decision parameter (non-categorical variable), it is possible to obtain and compare the respective relative operating characteristic curves (aROC) (Swets, 1988). However, aROC is not possible for systems without a single decision parameter. Thus, precision (P) and recall (R) are reduced to a single F-score, defined as  $2PR/(P+R)$ , which is the harmonic mean of precision and recall (Natarajan et al., 2005;

Tsai et al., 2006), and is always between 0 and 1 where 1 means that the system produces neither FP or FN. F-score assigns the same weight to both precision and recall, that is, both are equally important. A more general form of F-score allows for different weight to be assigned to precision and recall (Natarajan et al., 2005). Hirschman et al. (2002) proposed a variant of F-score, simple matching coefficient (SMC), which is defined as  $TP/(TP+FN+FP)$ .

### 1.5.9. Challenges of Biomedical Literature Analysis

The challenge for the field of biomedical literature analysis is to manage and process large amount of literature. The rate of publication of literature has exceeded the capacity to manually review it. Therefore, there is an increasing need for IE in this post-genomic era where the focus of biomedical research is shifting from the study of individual proteins and genes to entire biological systems.

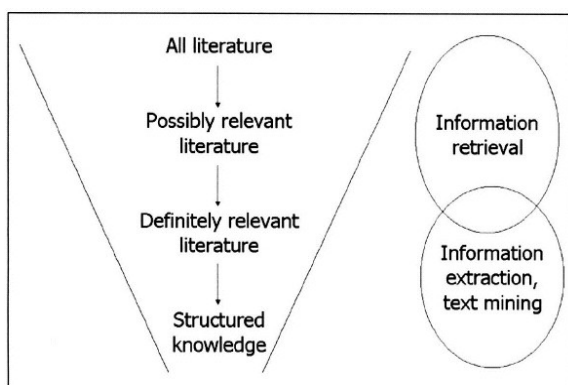


Figure 1e. The funnel of knowledge-based information. Demonstrating how the structuring and understanding of knowledge progresses from all biomedical literature becomes more refined and relevant. With larger amounts of literature, information retrieval techniques are essential and once the relevant literature has been defined, information extraction and text mining are required (Hersh, 2005).

Biomedical literature analysis (see Figure 1d) consists of getting source text from text repository (information retrieval; IR), finding information of interests within the text (information extraction; IE) and in the process, may require the recognition of entities like proteins (named entity recognition; NER) and evaluation of abbreviations (abbreviations recognition; AR). Extracted information may either be deposited into specialized databases as structured knowledge or may support knowledge discovery or support hypothesis generation by text mining (TM). Testing and evaluation of each step in the analysis requires the presence of a defined data set (corpus) for certain metrics, like recall measure, to be estimated. Alternatively, evaluation can be done by comparing with a suitable, existing database. Inherent in this process is the definition of an



appropriate evaluation strategy to allow for comparisons across similar systems.

The main repository of biomedical textual data is PubMed (MedLine) which provided only one means for searching relevant information. Although tools such as Textpresso (Muller et al., 2004) and MedMiner (Tanabe et al., 1999) represented an alternative, they are essentially bootstrapping on PubMed's IR. Despite its universal use, neither the implementation details (source code) of PubMed's IR engine is known (Simon et al., 2004; Wheeler et al., 2006) nor is the engine thoroughly evaluated for precision and recall. All research to date that retrieved data from PubMed as source had assumed that the data is pristine but this is neither possible nor verified. As a result, certain research questions in biomedical literature analysis which depended on the accuracy of PubMed, such as, what does the collective research knowledge to date tells us about the proteomic and metabolomic differences between mouse and rat, is fundamentally impossible at this stage.

Information extraction is heavily reliant on precise NER (Krauthammer and Nenadic, 2004) and AR. In spite of on-going debate as to the required performance of automated NER (de Bruijn and Martin, 2002) and AR systems before being useful in biomedical literature analysis, it is clear that near-human precision is unlikely in the near future. As a result of non-optimal performance (human performance is assumed optimal), IE is generally limited to *a priori* approach, that is, the user has to know the list of proteins or entities whose relationship he/she is interested in, as opposed to an *a posteriori* approach, such as finding relationships of possibly new entities or entities that are unknown to the user at time of search. Drawing analogies from genomic studies, *a priori* approach is analogous to microarray technology (the spots on the chip are pre-determined before actual experimentation) while *a posteriori* approach is likened to Massive Parallel Signature Sequencing (MPSS) (Stolovitzky et al., 2005). Nevertheless, current or in future, improved NER and AR systems can be of great use in assisting human curators to assemble a near complete dictionary (Shi and Campagne, 2005), such as ADAM (Zhou et al., 2006). Systems that learn from human curation efforts, in attempt to improve its performance, had recently surfaced (Rodriguez-Esteban et al., 2006). A system that performed automated curation of extracted interactions from text at the graph-level has also emerged (Rzhetsky et al., 2006). Another similar problem is

the need to recognize variants of the same name, commonly known as gene normalization (Hakenberg et al., 2008)

Relationships of paired entities by co-occurrence statistics usually requires large volumes of initial text as the term frequency of each entity (number of occurrences of a term divided by the total number of documents) is low, presumably less than 5% of the document set (corpus). This is likely to restrict its use on “small” corpus of less than 100000 and PubGene had used more than 10 million abstracts to generate its gene network (Jenssen et al., 2001). There are three advantages of co-occurrence methods when compared to NLP. Firstly, co-occurrence is basically statistical correlation and is easier to understand by more biologists than NLP techniques. Secondly, most NLP systems work at the sentence level; thus, cannot extract relationships either spanning more than one sentence or in complex sentences, where information may be split across multiple SVO tuples. On the other hand, co-occurrence can be easily deployed on different granularity of text. But it is necessary to note that the fundamental assumption of co-occurrence is independent observations, which is assumable for abstracts as full text usually refer to prior papers (in its introduction and discussion), thus, independent observations cannot be assumed. Lastly, it is known that co-occurrence methods generally has a higher recall but lower precision as compared to NLP means (Ding et al., 2002; Wren and Garner, 2004), and given that IE by NLP generally suffer from poor recall, it may be possible to improve the overall performance (improving recall substantially while suffering a small decline in precision) by simultaneously employing both co-occurrence and NLP (He et al., 2009).

As previously described, biomedical IE is driven by the Specialist (biomedical text are highly domain-specific and require specially developed NLP tools) and Generalist (generic or existing non-biomedically focused NLP tools can be adapted for biomedical use) schools of thought. However, both directions require either formulation of rules and templates or re-training parts of the system. Both tasks are manually intensive that requires manually tagged corpus (Jensen et al., 2006). Furthermore, there is no certainty in a better system and combining existing tools may be synergistic (Miyao et al., 2009). These approaches generally do not fall within the expertise of biologists which are the very people using the systems (Zhou and He, 2008). Although work by Grover (Grover

et al., 2002; Grover et al., 2003) suggested that native generic NLP tools may be used in biomedical text, to date, there are no study examining the un-modified use of a generic NLP system for biomedical literature analysis or comparing the native deployment of a generic NLP system to modified versions of the system with respect to biomedical IE. It has generally been assumed that some modifications must be made to generic NLP systems, especially the POS tagger, for it to be used on biomedical text, but this assumption has not been tested formally.

To make evaluation across systems, all biomedical literature analysis systems require some form of evaluation, either as performance metrics, such as precision and recall, or as statistical confidence of results (like in BLAST). However, this approach faces a number of challenges. Firstly, evaluation by performance often requires tagged corpora which are in severe shortage (Leser and Hakenberg, 2005) and most available corpora do not provide programmatic tools to use them readily. Hence, developers have to implement access routines in a particular computer programming language for each new corpus. Secondly, the current evaluation of individual systems make it difficult for comparison even though performance metrics may be known. This is due to different approaches used to obtain the performance metrics. For example, GENIES (Friedman et al., 2001) was evaluated using only one paper; BioRAT (David et al., 2004) was evaluated against an existing database, DIP (Salwinski et al., 2004); E3Miner (Lee et al., 2008) was evaluated against 47 abstracts. In order to be statistically sound, all systems should preferentially be evaluated against a common set of data, which was accomplished in challenges, such as TREC ([trec.nist.gov/](http://trec.nist.gov/)) (Voorhees and Buckland, 2005), BioCreative and LLL ([www.cs.york.ac.uk/aig/lll/](http://www.cs.york.ac.uk/aig/lll/)) (Cussens and Dzeroski, 2000). Alternatively, a common dataset can be established for communal use (Cano et al., 2009; Pyysalo et al., 2008; Zhou and He, 2008), like that of UCI Machine Learning Repository (Newman et al., 1998).

One of the main reasons this technology is slowly adopted by biologists is because they are not trained in computer science to integrate the tools effectively (Kano et al, 2008). Therefore, it is necessary to present clear benefits of using these tools (Altman et al., 2008; Cohen and Hersh, 2005; Muller et al., 2007; Krallinger et al., 2008b; Zhou and He, 2008). It is almost a tradition in data analysis and mining to create a system that

allows users to set their own parameters in accordance to the task at hand or to evaluate a system independent of meeting user needs (Cohen and Hersh, 2005). However, the biologists using the system, who are generally clueless about the nature of each parameters, faces a daunting task of setting these parameters. Therefore, it is necessary to involve the biologists in the process of creating new tools or adapting or aggregating existing tools to help biomedical researcher to solve real world problems (Cohen and Hersh, 2005; Caporaso et al., 2008; Roberts and Hayes, 2008). Hence, a recent trend is to use literature analysis to provide and update evidence data for Gene Ontology annotations (Roberts, 2006; Couto et al., 2006; Lussier et al., 2006; Natarajan and Ganapathy, 2007; Cakmak and Ozsoyoglu, 2008; Jin et al., 2008; Kim et al., 2008; Yang et al., 2008b), combining literature analysis with ontology information for query answering (Abulaish and Dey, 2007), extracting concepts from text (Baumgartner et al., 2008) or to access the information needs of specific areas of research (Roberts and Hayes, 2008). At the same time, literature analysis has also been used to group genes based on their functions (Heinrich et al., 2008), extracting medically important terms from text (Jimeno et al., 2008) and further the development of new ontologies (Spasic et al., 2008).

The golden age of biomedical literature analysis of 1998 to 2004 had left us with a number of disjoint sets of tools: systems for specific purposes in the process, such as MedPost; systems for specific biological purposes, like microGENIES; various ontologies and lexicons, like Textpresso Ontology, GENIA Ontology; visualization tools, etc. Although it seems that technology transfer from more traditional fields, such as computational linguistics for understanding general text, had reached its maximum in that period, creative use of these techniques, picking up and re-structuring the pieces left behind, and targeting the resulting systems to the actual needs of biologists, could bring forth the next golden age of biomedical literature analysis.

### ***1.6. Microarray Analyses Assisted by Biomedical Literature Analyses***

Although a number of systems had been developed with specific uses in mind, such as microGENIE (Korotkiy et al., 2004) and MILANO (Rubinstein and Simon, 2005), there was little evidence suggesting these tools were actively in use or not obvious in

resulting publications, which might suggest that adoption of biomedical literature analysis by biologists had been slow (Cohen and Hersh, 2005).

The most relevant studies were that of PubGene (Jenssen et al., 2001) and CoPub Miner (Alako et al., 2005) which demonstrated that co-occurrence statistics correlates well to microarray results. Using a compiled knowledge base from literature, Gutierrez-Rios et al. (2003) had demonstrated consistency between the literature and microarray profiles in *Escherichia coli*. Karopka et al. (2004) demonstrated a preliminary use of a biomedical NLP system for microarray result analysis. Natarajan et al. (2006) combined NLP with gene expression analysis to examine a relationship between sphingosine-1-phosphate and glioblastoma cell line. A combined literature mining by co-occurrence and microarray analysis (LMMA) approach (Li et al., 2006b) had been used to construct biological networks that are more reliable than co-occurrence alone in handling multiple levels of KEGG genes, KEGG Orthology and KEGG pathways. Topinka and Shyu (2006) combined NLP with structure-based information, sub-cellular localization and evolutionary information to predict cancer interaction networks. Ma'ayan et al. (2006) attempted to understand Down's Syndrome using text analysis and experimental data. Li et al. (2007) developed a framework for assisting the understanding of p53 with text analysis means. Gajendran et al. (2007) discovered novel genes related to bone biology using genomics and text analysis methods. Hsu et al. (2007) incorporated information from 8 different sources to study hepatocellular carcinoma. Despite so, all of these systems were still in research and development phase and little had been done in terms of user evaluations (Cohen and Hersh, 2005) and there are no studies to date combining statistical literature analysis, computational linguistics analysis of the literature and microarray profiles.

### **1.7. Tools for Visualizing Large Graphs**

With the amount of literature available and the complexity of protein-protein interaction networks, listing the results in table format is usually not desirable and difficult to navigate. Graphs had been slowly adopted as a means to summarize and present results (Han et al., 2004). A number of systems had used graphs as an alternative means (from tables) to present results to users, such as GeneScene (Leroy and Chen, 2002) and VisAnt (Hu et al., 2005a).

Graphviz (Gansner and North, 1999) had been used in some studies as a result visualization tool (Bodenreider and Mitchell, 2003; Wong, 2001). While Graphviz, being command driven, is relatively easy to use in the beginning, advanced uses might require some learning. At the same time, visual editing is impossible in Graphviz. On the other hand, Cytoscape (Shannon et al., 2003) allowed for visual editing of graphs which might be more suited for biologists. At the same time, several plugins had been developed for Cytoscape (Albrecht et al., 2005; Maere et al., 2005; Vlasblom et al., 2006, Garcia et al., 2007, Ferro et al., 2007, Barsky et al., 2007)

## ***1.8. Objectives and Organization of this Thesis***

There are two main objectives to this thesis. The first objective is to examine the possibility of combining literature and genomic analysis to elucidate potentially novel hypotheses for further research into lactation biology. Literature analysis can be broadly classified into statistics-based (co-occurrence) or linguistic-based (natural language processing). Chapter 3 examines the use of a generic natural language processor to extract protein-protein interactions from published abstracts. Chapter 3 attempts to resolve the contradiction presented by Chapter 2 where comparable performance in the extraction protein-protein interactions is achieved when using a generic text processor compared to that of using specialized tools. Chapter 4 examines the use of co-occurrence statistic as a sieve of scientific knowledge to filter correlated gene expression profiles from microarray data set for potentially novel hypotheses of functionally related proteins. Chapter 4 also examines the possibility of annotating the types of interactions to pairs of proteins from co-occurrence statistics.

The second objective is to evaluate the strengths and limitations of the murine mammary explant culture for the study and understanding of murine lactogenesis. The underlying question to this objective is whether the mouse mammary explant culture is a good model or representation to study mouse lactogenesis. Chapter 5 compares the transcriptional profiles of mouse mammary explants before and after IFP induction and that of in vivo mouse mammary glands before and after parturition to address the question of whether the IFP-induced mammary explants mimic the in vivo lactogenic

response.

Chapter 6 concludes this thesis with a general discussion of the findings of this thesis, its limitations and possible future work.

## Chapter Two

# Reconstruction of Protein-Protein Interaction Pathways by Mining Subject-Verb-Objects Intermediates

### 2.1 Introduction

PubMed currently indexes more than 16 million papers with about one million papers and 1.2 million added in the years 2005 and 2006 respectively. A simple keyword search in PubMed showed that nearly 900 thousand papers on mouse and more than 1.3 million papers on rat research had been indexed in PubMed to date, and in the last four years, more than 150 thousand papers have been published on each of mouse and rat research. This trend of increased volume of research papers indexed in PubMed over the last 10 years makes it difficult for researchers to maintain an active and productive assessment of relevant literature. Information extraction (IE) has been used as a tool to analyze biological text to derive assertions on specific biological domains (Rebholz-Schuhmann et al., 2005), such as protein phosphorylation (Hu et al., 2005) or entity interactions (Abulaish and Dey, 2007).

A number of IE tools used for mining information from biological text can be classified according to their capacity for general application or tools that considers biological text as specialized text requiring domain-specific tools to process them. This has led to the development of specialized part-of-speech (POS) tag sets (such as SPECIALIST (National Library of Medicine, 2003)), POS taggers (such as MedPost (Smith et al., 2004)), ontologies (Daraselia et al., 2004), text processors (such as MedLEE (Friedman et al., 1994)), and full IE systems, such as GENIES (Friedman et al., 2001), MedScan (Novichkova et al., 2003), MeKE (Chiang and Yu, 2003), Arizona Relation Parser (Daniel et al., 2004), and GIS (Chiang et al., 2004). On the other hand, an alternative approach assumes that biological text are not specialized enough to warrant re-development of tools but adaptation of existing or generic tools will suffice. To this end, BioRAT (David et al., 2004) had modified GATE (Cunningham, 2000), MedTAKMI



(Uramoto et al., 2004) had modified TAKMI (Nasukawa and Nagono, 2001), originally used in call centres, Santos (Santos et al., 2005) had used Link grammar parser (Sleator and Temperley, 1991).

Although both systems demonstrated similar performance, either developing these systems or modifying existing systems were time consuming (Jensen et al., 2006). Although work by Grover (Grover et al., 2002) suggested that native generic tools may be used for biological text, a recent review had highlighted successful uses of a generic text processing system, MontyLingua (Eslick and Liu, 2005; Liu, 2004), for a number of purposes (Ling, 2006). For example, MontyLingua has been used to process published economics papers for concept extraction (van Eck and van den Berg, 2005). The need to modify generic text processors had not been formally examined and the question of whether an un-modified, generic text processor can be used in biological text analysis with comparable performance, remains to be assessed.

In this study, we evaluated a native, generic text processing system, MontyLingua (Liu and Singh, 2004), in a two-layered generalization-specialization architecture (Novichkova et al., 2003) where the generalization layer processes biological text into an intermediate knowledge representation for the specialization layer to extract genic or entity-entity interactions. This system demonstrated 86.1% precision using Learning Logic in Languages 2005 evaluation data (Cussens, 2005), 88.1% and 90.7% precisions in extracting protein-protein binding and activation interactions respectively. Our results were comparable to previous work which modified generic text processing systems which reported precision ranging from 53% (Malik et al., 2006) to 84% (Chiang et al., 2004), suggesting this modification may not improve the efficiency of information retrieval.

## **2.2 System Description**

We have developed a biological text mining system, known as Muscorian, for mining protein-protein inter-relationships in the form of subject-relation-object (for example, protein X bind protein Y) assertions. Muscorian is implemented as a 3-module sequential system of entity normalization, text analysis, and protein-protein binding

finding, as shown in Figure 2a. It is available for academic and non-profit users through <http://www.sourceforge.net/projects/muscorian>.

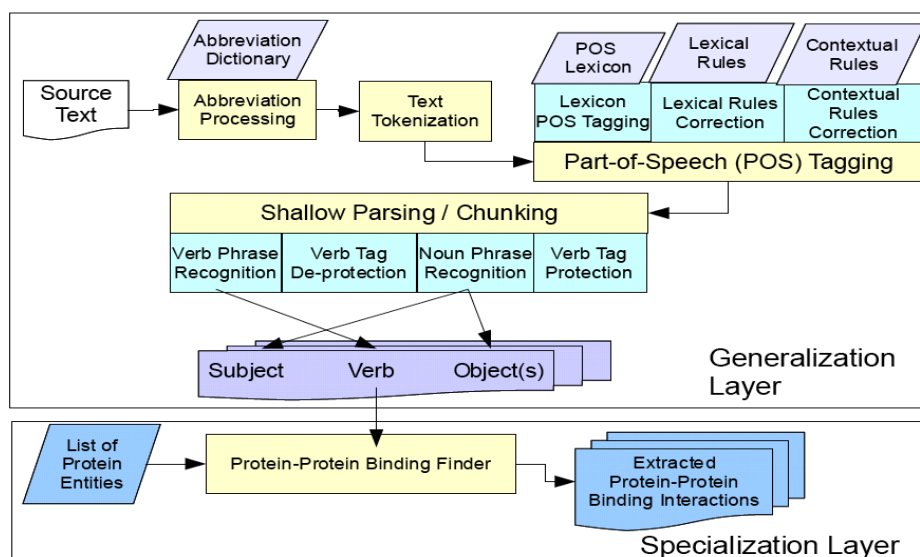


Figure 2a. Schematic Diagram Illustrating the Operations of Muscorian

## 2.2.1 Entity Normalization

Entity normalization is the substitution of the long form of either a biological or chemical term with its abbreviated form. This is essential to correct part-of-speech tagging errors which are common in biological text due to multi-worded nouns. For example, the protein name “phosphatase and tensin homolog deleted on chromosome 10” has to be recognized as a single noun and not a phrase. In this study, we attempt to mine protein-protein interactions and consolidate this knowledge to produce a map. Therefore, the naming convention of the protein entities must be standardized to allow for matching. However, this is not the case for biological text and synonymous protein names exist for virtually every protein. For example, “MAP kinase kinase”, “MAPKK”, “MEK” and “MAPK/Erk kinase” referred to the same protein. Both of these problems could be either resolved or minimized by reducing multi-worded nouns into their abbreviated forms.

A dictionary-based approach was used for entity normalization to a high level of accuracy and consistency. The dictionary was assembled as follows: firstly, a set of 25000 abstracts from PubMed was used to interrogate Stanford University's BioNLP

server (Chang et al., 2002) to obtain a list of long forms with its abbreviations and a calculated score. Secondly, only results with the score of more than 0.88 were retained as it is an inflection point of ROC graph (Chang et al., 2002), which is a good balance between obtaining the most information while reducing curation efforts. Lastly, the set of long form and its abbreviations was manually curated with the help of domain experts.

The domain experts curated dictionary of long forms and its abbreviated term was used to construct a regular expression engine for the process of recognition of the long form of a biological or chemical term and substituting it with its corresponding abbreviated form.

### **2.2.2 Text Analysis**

Entity normalized abstracts were then analyzed textually by an un-modified text processing engine, MontyLingua (Eslick and Liu, 2005), where they were tokenized, part-of-speech tagged, chunked, stemmed and processed into a set of assertions in the form of 3-element subject-verb-object(s) (SVO) tuple, or more generally, subject-relation-object(s) tuple. Therefore, a sequential pattern of words which formed an abstract was transformed through a series of pattern recognition into a set of structurally-definable assertions.

Before part-of-speech tagging is possible, an abstract made up of one or more sentences had to be separated into individual sentences. This is done by regular expression recognition of sentence delimiters, such as full-stop, ellipse, exclamation mark and question mark, at the end of a word (regular expression: ([?!]+[.][.]+)\$) with an exception of acronyms. Acronyms, which are commonly represented with a full-stop, for example “Dr.”, are not denoted as the end of a sentence and were generally prevented by an enumeration of common acronyms.

Individual sentences were then separated into constituent words and punctuations by a process known as tokenization. Tokenization, which is essential to atomize a sentence into atomic syntactic building blocks, is generally a simple process of splitting of an

English sentence in words using whitespaces in the sentence, resulting in a list of tokens (words). However, there were three problems which were corrected by examining each token. Firstly, punctuations are crucial in understand a written English sentence, but typographically a punctuation is usually joined to the presiding word. Hence, punctuation separation from the presiding word is necessary. However, it resulted in incorrect tokenization with respect to acronyms and decimal numbers. For example, "... an appt. for ..." will be tokenized to "... an appt . for ..." and "\$4.20" will be "\$ 4 . 20". This problem was prevented by pre-defining acronyms and using regular expressions, such as " $^{\wedge}[\$][0-9]\{1,3\}[\.][0-9][0-9](?[\.])?\$$ ". Lastly, common abbreviated words, such as "don't", were expanded into two tokens of "do" and "n't". Despite the above error correction measures, certain text such as mathematical equations, which might be used to describe enzyme kinetics in biological text, will not be tokenized correctly. In spite of this limitation, the described tokenization scheme is still appropriate as extraction of enzyme kinetics or mathematical representations are not the aims of this study.

Each of the tokens (words and punctuations) in a tokenized sentence is then tagged using Penn TreeBank Tag Set (Marcus et al., 1993) by a Brill Tagger, trained on Wall Street Journal and Brown corpora, which operates in two phases. Using a lexicon, containing the likely tag for each word, each word is tagged. This is followed by a phase of correction using lexical and contextual rules, which were learnt using training with a tagged corpora, in this case, Wall Street Journal and Brown corpora. Lexical rules uses a combination of preceding tag and prefix or suffix of the token (word) in question. For example, the rule "NN ing fhassuf 3 VBG" defines that if the current token is tagged as a noun (NN) and has a 3-character suffix of "ing", then the tag should be a verb (VBG). On the other hand, contextual rules uses only the preceding or proceeding tags and hence, must be applied after lexical rules for effectiveness. The contextual rule "RB JJ NEXTTAG NN" defines that an abverbial tag (RB) should be changed to an adjective (JJ) if the next token was tagged as a noun (NN). A table of Penn Treebank Tag Set (Marcus et al., 1993) without punctuation tags is given in Table 2a.

By tagging, the complexity of an English sentence (ie, the number of ways an English sentence can be grammatically constructed with virtually unlimited words and unlimited ideas) was collapsed into a sequence of part-of-speech tags, in this case, Penn TreeBank

Tag Set (Marcus et al., 1993), with only about 40 tags. Therefore, tagging reduced the large number of English words to about 40 “words” or tags.

Table 2a. Penn Treebank Tag Set without Punctuation Tags (Adapted from (Marcus et al., 1993))

Tag	Description	Tag	Description
CC	Coordinating conjunction	PRP\$	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determinant	RBR	Adverb, comparative
EX	Existential <i>there</i>	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	to
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBN	Verb, past participle
NN	Noun, singular or mass	VBG	Verb, gerund or present participle
NNS	Noun, plural	VBP	Verb, non-3 <sup>rd</sup> person singular present
NNP	Proper noun, singular	VBZ	Verb, 3 <sup>rd</sup> person singular present
NNPS	Proper noun, plural	WDT	Wh-determiner
PDT	Predeterminer	WP	Wh-pronoun
POS	Possessive ending	WP\$	Possessive wh-pronoun
PRP	Personal pronoun	WRB	Wh-adverb

Generally, an English sentence is composed of a noun phrase, a verb, and a verb phrase, where the verb phrase may be reduced into more noun phrases, verbs, and verb phrases. More precisely, the English language is an example of subject-verb-object typology structure, which accounts for 75% of all languages in the world (Crystal, 1997). This concept of English sentence structure is used to process a tagged sentence into higher-order structures of phrases by a process of chunking, which is a precursor to the extraction of semantic relationships of nouns into SVO structure. Using only the

sequence of tags, chunking was performed as a recursive 4-step process: protecting verbs, recognition of noun phrases, unprotecting verbs and recognition of verb phrases. Firstly, verb tags (VBD, VBG and VBN) were protected by suffixing the tags. The main purpose was to prevent interference in recognizing noun phrases. Secondly, noun phrases were recognized by the following regular expression pattern of tags:

```
(( ((PDT )?(DT |PRP[$] |WDT |WP[$] ) (VBG |VBD |VBN |JJ |JJR |JJS |, |CC |NN |NNS |NNP |NNPS |CD )*(NN |NNS |NNP |NNPS |CD )+ ) ((PDT )?(JJ |JJR |JJS |, |CC |NN |NNS |NNP |NNPS |CD )*(NN |NNS |NNP |NNPS |CD )+ ) |EX |PRP |WP |WDT ) POS )? (( (PDT )?(DT |PRP[$] |WDT |WP[$] ) (VBG |VBD |VBN |JJ |JJR |JJS |, |CC |NN |NNS |NNP |NNPS |CD )*(NN |NNS |NNP |NNPS |CD )+ ) ((PDT )?(JJ |JJR |JJS |, |CC |NN |NNS |NNP |NNPS |CD )*(NN |NNS |NNP |NNPS |CD )+ ) |EX |PRP |WP |WDT )
```

Thirdly, the protected verb tags in the first step were de-protected by removing the suffix appended onto the tags. Lastly, verb phrases were recognized by the following regular expression:

```
((RB |RBR |RBS |WRB )*(MD )?(RB |RBR |RBS |WRB )*(VB |VBD |VBG |VBN |VBP |VBZ ) (VB |VBD |VBG |VBN |VBP |VBZ |RB |RBR |RBS |WRB )*(RP )?(TO (RB )*(VB |VBN ) (RP )?)?)
```

After chunking, each word (token) was stemmed into its root or infinite form. Firstly, each word was matched against a set of rules for specific stemming. For example, the rule “dehydrogenised verb dehydrogenate” defines that if the word “dehydrogenised” was tagged as a verb (VBD, VBG and VBN tags), it would be stemmed into “dehydrogenate”. Similarly, the words “binds”, “binding” and “bounded” were stemmed to “bind”. Secondly, irregular words which could not be stemmed by removal of prefixes and suffixes, such as “calves” and “cervices”, were stemmed by a pre-defined dictionary. Lastly, stemming was done by simple removal of prefixes or suffixes from the word based on a list of common prefixes or suffixes. For example, “regards” and “regarding” were both stemmed into “regard”.

Given the general nature of an English sentence is an aggregation of noun phrase, a verb, and a verb phrase, where the verb phrase may be reduced into more noun phrases,

verbs, and verb phrases, each verb phrase may be taken as a sentence by itself. This allowed for recursive processing of a chunked-stemmed sentence into SVO(s) by a 3-step process. Firstly, the first terminal noun phrase, delimited by “(NX” and “NX)” was taken as the subject noun. Secondly, proceeding from the first terminal noun phrase, the first terminal verb would be taken as the verb in the SVO. Lastly, the rest of the phrase was scanned for terminal noun phrases and would be taken as the object(s). The recursive nature of SVO extraction also meant that the subject, verb, and object(s) will be contiguous, which had been demonstrated to have better precision than non-contiguous SVOs (Masseroli et al., 2006).

### **2.2.3 Protein-Protein Binding Finding**

The protein-protein binding finder module is a data miner for protein-protein binding interaction assertions from the entire set of subject-relation-object (SVO) assertions from the text analysis process using apriori knowledge. That is, the set of proteins of interest must be known, in contrast to an attempt to uncover new protein entities, and their binding relationships with other protein entities, that were not known to the researcher.

Protein-protein binding assertions were extracted in a three step process. Firstly, a set of SVOs was isolated by the presence of the term “bind” in the verb clause resulting in a set of “bind-SVOs” assertions. Non-infinite forms of “bind” (such as, “binding” and “binds”) were not used as verbs were stemmed into their infinite forms during text processing. Secondly, the set of bind-SVOs were further characterized for the presence of protein entities in both subject and object clauses by comparing with the desired list of protein entities. A pairwise isolation of bind-SVOs for protein entities resulted in a set of bind-SVOs, “entity-bind-SVOs”, containing SVOs describing binding relationship between the protein entities. Lastly, entity-bind-SVOs were cleaned so that the subject and object clauses only contains protein entities. For example, “MAPK in the cytoplasm” in the object clause will be reduced to just the entity name “MAPK”, the full subject and object clauses could be used in other information extraction tasks, such as determining protein localization, but is not explored in this study. This step is required to allow for the construction of network graphs, such as using Graphviz,

without reference to the list of protein names during construction. Given that `protein_entities` is the list of desired proteins, table `SVO` contains the SVO output from MontyLingua and table `entity_bind_SVO` contains the isolated and cleaned SVOs, the pseudocode for Protein-Protein Binding Finding module is given as:

```

for subject_protein in protein_entities1 to n
  for object_protein in protein_entities1 to n
    insert (pmid, subject_protein, object_protein) into entity_bind_SVO
      from select pmid
      from (select * from SVO where verb = 'bind')
      where subject is containing subject_protein
      and object is containing object_protein

```

## 2.3 Experimental Results

Four experiments were carried out to evaluate the performance of Muscorian and demonstrate the flexibility of the two-layered generalization-specialization approach in constructing systems that could be readily be adapted to related problems. The results are summarized in Table 2b.

Table 2b. Summary of the Experimental Results Comparing the Precision and Recall Measures.

	<i><b>LLL05 Directional</b></i>	<i><b>LLL05 Un-directional</b></i>	<i><b>Protein-Protein Binding</b></i>	<i><b>Protein-Protein Activation</b></i>
Precision	55.8%	86.1%	88.1%	90.7%
Recall	19.8%	30.7%	Not measured	Not measured

### 2.3.1 Benchmarking Muscorian Performance

The performance of Muscorian, in terms of precision and recall, could only be evaluated using a defined data set with known results. For such purpose, the data set for Learning Languages in Logic 2005 (LLL05) (Cussens, 2005) was used to benchmark Muscorian on genic interactions, which is a superset of protein-protein binding interactions. LLL05 had defined a genic interaction as an interaction between 2 entities (agent and target) but the nature of interaction was not considered under the challenge task. LLL05 provided a



list of protein entities found in the data set, which was used to filter subject-relation-object assertions from text analysis (MontyLingua) output where both subject and object contained protein entities in the given list. The filtered list of assertions was evaluated for precision and recall, which was found to be 55.6% and 19.8% respectively.

LLL05 required that the agent and target (subject and object) to be in the correct direction, making it a vector quality. However, this requirement was not biologically significant to protein-protein binding interactions, which is scalar. For example, “X binds to Y” and “Y binds to X” have no biological difference. Hence, this requirement of directionality was eliminated and the precision and recall was 86.1% and 30.7% respectively.

### **2.3.2 Verifying Protein-Protein Binding Interactions**

Precision of Muscorian for mining protein-protein binding interactions from published abstracts was evaluated by manual verification of a sample of assertions (n=135) yielded by the protein-protein binding finder module against the original abstracts. Each of the sampled assertions was assumed to be atomic, in the form of “X binds Y”. In cases where there were more than one target, such as “X binds Y and Z”, they would be reduced to atomic assertions. In this case, “X binds Y and Z” would be reduced to 2 assertions, “X bind Y” and “X bind Z”. These were then checked with the original abstract, traceable by the PubMed IDs, and precision was measured as the ratio of the number of correct assertions to the number of sampled atomic assertions (which is 135). A 95% confidence interval was estimated by bootstrapping (re-sampling with replacement) (Efron and Tibshirani, 1986) of the manual verification results. Our results suggested a precision of 88.1%, with a 95% confidence interval between 82.4% to 93.7%.

An IE trial was performed using the Protein-Protein Binding Finding module to search for the binding partners of CREB and insulin receptor and a sample network diagram of the results are shown in Figure 2b and 2c respectively.

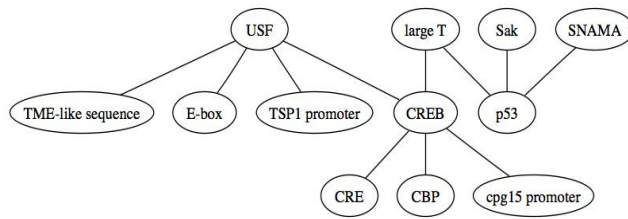


Figure 2b. Preliminary Protein Binding Network of CREB

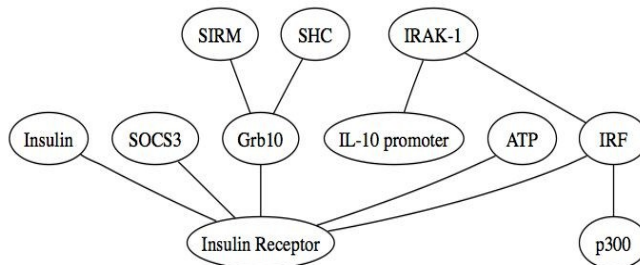


Figure 2c. Preliminary Protein Binding Network of Insulin Receptor

### 2.3.3 Large Scale Mining of Protein-Protein Binding Interactions

A large scale mining of protein-protein binding interactions was carried out using all of the PubMed abstracts on mouse (about 860000 abstracts), which were obtained using “mouse” as the keyword for searches, with a predefined set of about 3500 abbreviated protein entities as the list of proteins of interest (available from [https://muscorian.svn.sourceforge.net/svnroot/muscorian/muscorian-data/protein\\_accession.csv](https://muscorian.svn.sourceforge.net/svnroot/muscorian/muscorian-data/protein_accession.csv)). In this experiment, the primary aim was to apply Muscorian to a large data set and the secondary aim was to look for multiple occurrences of the same interactions as multiple occurrences might greatly improve precision confidence.

For example, given our lower confidence estimate that the precision of Muscorian with respect to mining protein-protein binding interactions is 82%, which means that every binding assertion has an 18% likelihood of not having a corresponding representation in the published abstracts. However, if 2 abstracts yielded the same binding assertion, the probability of both being wrong was reduced to 3.2% ( $0.18^2$ ), and the corresponding probability that at least one of the 2 assertions was correctly represented was 96.8% ( $1 - 0.18^2$ ). The more times the same assertion was extracted from multiple sources text (abstracts), the higher the possibility that the mined interaction was represented at least once in the set of abstracts. For example, if 5 abstracts yielded the same assertion, the possibility that at least one of the 5 assertions was correctly represented would be 99.98% ( $1 - 0.18^5$ ).

Our experiment mined a total of 9803 unique protein-protein binding interactions, of which 7049 binding interactions were from one abstract (P=82%), 1297 binding interactions were from two abstracts (P=96.8%), 516 binding interactions were from three abstracts (P=99.4%), 235 binding interactions were from four abstracts (P=99.9%), 164 binding interactions were from five abstracts (P=99.98%), 105 binding interactions were from six abstracts (P=99.997%), 69 binding interactions were from seven abstracts (P=99.9993%), 398 binding interactions were from more than seven abstracts (P>99.9993%).

### **2.3.4 Pilot Study - Protein-Protein Activation Interactions**

In order to demonstrate the adaptability of our proposed two-layered model, a small pilot study for mining protein-protein activation interactions was carried out. For this study, the protein-protein binding finder module, the data mining module for mining protein-protein binding interaction, was replaced with a protein-protein activation finder module.

The protein-protein activation finder was semantically similar to the original protein-protein binding finder module as described in Section 2.3.3 previously. The only difference was that raw assertion output from MontyLingua was filtered for activation-related assertions, instead of binding-related assertions, before analysis for the presence of protein names in both subject and object nouns from a pre-defined list of proteins of interest. For example, by modifying the Protein-Protein Binding Finding module to look for the verb 'activate' instead of 'bind', it can then be used for mining protein-protein activation interactions. A trial was done for insulin activation and a subgraph is illustrated in Figure 2d below.

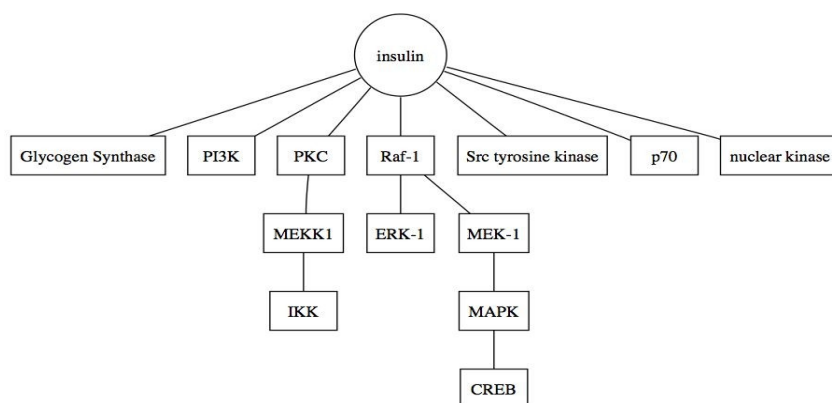


Figure 2d. Preliminary Protein Activation Network of Insulin

The precision measure of Muscorian for mining protein-protein activation interactions was calculated using identical means as described for protein-protein binding interactions. Using a sample of 85 atomic assertions, the precision of Muscorian for mining protein-protein activation interactions was estimated to be 90.7%, with a 95% confidence interval of precision between 84.7% to 96.4% by bootstrapping (Efron and Tibshirani, 1986).

## 2.4 Discussion

New research articles in gene expression regulation networks, protein-protein interactions and protein docking are emerging at a rate faster than what most biologists can manage to extract the data and generate working pathways. Information extraction technologies have been successfully used to process research text and automate fact extraction (Abulaish and Dey, 2007). Previous studies in biological text mining have developed specialized text processing tools and adapted generic tools to relatively good performance of more than 80% in precision (Chiang and Yu, 2004; Daraselia et al., 2004; Jensen et al., 2006; Santos et al., 2005). However, either specialized tool development or modifying existing tools often require much effort (Jensen et al., 2006). The need to modify existing tools has not been formally tested and the possibility of using an un-modified generic text processor for biological text for the purpose of extracting protein-protein interaction remains unresolved. Using a two-layered approach (Novichkova et al., 2003) of generalizing biological text into a structured intermediate

form, followed by specialized data mining, we present Muscorian, which uses MontyLingua natively in the generalized layer, as a tool for extracting either protein-protein or genic interactions from about 860000 published biological abstracts.

Benchmarking Muscorian against LLL05, a tested data set, demonstrated a precision of 55.6%, which is about 5% higher than that reported in the conference and a recall of 19.7% is similar to that reported by other participants of LLL05 (Cussens, 2005). This may be due to the emphasis of LLL05 on F-measure, which is the harmonic mean of precision and recall, rather than putting more emphasis on precision. Nevertheless, this also suggested that Muscorian is able to perform text analysis for the purpose of extracting genic interactions effectively, which is comparable to specialized systems reported in LLL05. In addition, directionality of genic interactions was not a concern for protein-protein binding interactions as binding interaction is scalar rather than vector. By eliminating directionality of genic interactions, the precision and recall of Muscorian was 86.1% and 30.7% respectively. This suggested that Muscorian is a suitable tool for mining quality genic interactions from biological text compared to other tools reported in LLL05 (Cussens, 2005).

Our results on protein-protein binding and activation interactions show the insulin receptor binds to IL-10 promoter through IRF and IRAK-1, which is an important insulin receptor signalling pathway. In addition, our data shows insulin activates CREB via Raf-1, MEK-1 and MAPK, which is consistent with the MAP kinase pathway. Combining these data (Figures 2.2 and 2.4) indicated that insulin activates CREB via MAP kinase pathway, and CREB binds to cpg15 promoter in the nucleus. A simple keyword search on PubMed, using the term “cpg15 and insulin” (done on 30<sup>th</sup> of April, 2007), did not yield any results, suggesting that the effects of insulin on cpg15, also known as neuritin (Cappelletti et al., 2007), had not been studied thoroughly. This might also suggest limited knowledge shared between insulin investigators and cpg15 investigators as suggested by Don Swanson in his classical paper describing the links between fish oil and Raynaud's syndrome (Swanson, 1986). Neuritin is a relatively new research area with less than 20 papers published (as of 30<sup>th</sup> of April, 2007) and had been implicated as a lead for neural network re-establishment (Han et al., 2007), suggesting potential collaborations between endocrinologists and neurologists.

Our experiments in extracting two different forms of relations demonstrated that despite using specialized dictionaries in the generalized layer, it is still general to the extent that specific application (the type of relationships to extract) was not built into the generalized layer.

At the same time, these 2 experiments also illustrated the relative ease in re-targeting the system for extracting another form of relationship by modifying the specialized layer. The Protein-Protein Activation Finder module is a slight modification of the original Protein-Protein Binding Finder module where the original SQL statement that selects 'bind'-related SVOs from total SVOs, “*select \* from SVO where verb = 'bind'*”, was changed to “*select \* from SVO where verb = 'activate'*” to select for 'activation'-related SVOs from total SVOs. Hence, it is plausible that similar changes may suffice for extracting other relationships, such as 'inhibition'. This relative ease of re-targeting the system for extracting other relationships also demonstrated the robustness of the generalization layer, as implied by Novichkova et. al. (2003) – “*the adaptability of the system to related problems other than the problem the system was designed for*”.

Given large numbers of published abstracts, the performance of Muscorian on precision was comparable with published values of BioRAT (58.7%) (David et al., 2004), GIS (84%) (Chiang et al., 2004), Cooper and Kershenbaum (74%) (Cooper and Kershenbaum, 2005) and CONAN (53%) (Malik et al., 2006) while Muscorian's recall was comparable with published values of Arizona Relations Parser (35%) (Daniel et al., 2004) and Daraselia et. al. (21%) (Daraselia et al., 2004). Poor precision was considered unacceptable because incorrect information is more detrimental than missing information (1 - recall) when protein-protein binding interactions were used to support other biological analyses. Muscorian's mediocre recall of 30% (from LLL05 test set evaluation) could be supplemented by the fact that the same interaction could be mentioned or described by multiple abstracts; thus, the actual recall when tested on a large corpus may be higher. For example, 30% recall essentially means a loss of 70% of the information; however, if the same information (in this case, protein interactions) were mentioned in 3 or more abstracts, there is still a reasonable chance to believe that information from at least 1 of the 3 or more abstracts will be extracted. This is

supported by our results indicating that almost 30% (2754 of 9803) of binding interactions were extracted from more than one abstract.

Multiple isolation of 2754 binding interactions enabled a higher confidence that these interactions were correctly extracted with reference to the source literature. Based on this analysis, 2754 binding interactions could be assigned higher confidence based on their occurrences (Jenssen et al., 2001), in this case more than 95% chance of being correct based on literature. In addition, the number of multiple interaction occurrence varies inversely with the number of abstracts these interactions were found in is in line with expectation. Although this line of argument is based on the assumption that the appearance of protein names across abstracts were independent, it can be reasonably held as this study uses abstracts rather than full text – abstracts tends to describe what main results of the particular article while the introduction of a full text article tends to be a brief background review of the field. Hence, independence of protein names can be better assumed in abstracts than in full text articles.

An evaluation of a sample of atomic assertions (interactions) of binding and activation interactions between entities was performed by domain experts comparing the assertions with their source abstracts. Both approaches gave similar precision measures and are consistent with the evaluation using LLL05 test set. The ANOVA test demonstrated that there was no significant differences between these three precision measures. Taken together, these evaluations strongly suggested that Muscorian performed with precisions between 86-90% for genic (gene-protein and protein-protein) interactions, which was similar to that reported by studies either modifying existing tools (Santos et al., 2005) or developing specialized tools (Daraselia et al., 2004). This suggested that MontyLingua could be used natively (un-modified), with good precision, to process biological text into structured subject-verb-objects tuples which could be mined for protein interactions.

## Chapter 3

# Parts-of-Speech Tagger Errors Do Not Necessarily Degrade Accuracy in Extracting Information from Biomedical Text

### 3.1 Introduction

PubMed currently indexes more than 17.5 million papers that includes 1 million papers added in both 2006 and the first half of 2007. This trend of increased volume of research papers makes it difficult for researchers to maintain a productive assessment of relevant literature. Information extraction (IE) has been used as a tool to analyze biological text to derive assertions, such as entity interactions (Abulaish and Dey, 2007). To date, there has been a number of IE tools to extract entity interactions from published text, such as MedScan (Novichkova et al., 2003), Arizona Relation Parser (Daniel et al., 2004), BioRAT (David et al., 2004) and Santos et al. (2005).

A recent article by Ling et al. (2007) has classified entity interaction IE tools by whether tools are developed with biological text in mind or adapted generic tools for biological text. Ling et al. (2007) developed Muscorian, a tool to extract protein-protein interactions from text. They also demonstrated that a generic text analysis tool chain, MontyLingua (Liu and Singh, 2004; Ling, 2006), incorporated into a two-layered generic-specialized architecture as explained in MedScan (Novichkova et al. 2003), can give rise to comparable performance in entity interaction extraction compared to those IE systems that modified existing systems, such as BioRAT (David et al., 2004), Chilibot (Chen and Sharp, 2004) and Santos et al. (2005). One of the common features of both classes of tools defined by Ling et al. (2007) is the specialization of the part-of-speech (POS) tagger. For example, Arizona Relation Parser (Daniel et al., 2004) re-trained Brill tagger (Brill, 1995) and Chilibot (Chen and Sharp, 2004) re-trained TnT tagger (Brants, 2000). POS tagging is a process of assigning grammatical roles of each word and punctuation in the source sentence. This plays a critical role in subsequent text processing tasks, such as shallow parsing, where the sequence of POS tags was used



instead of the original sequence of words. At the same time, it was known that errors in POS tagging often results in misunderstanding of the sentence (Kodratoff et al., 2005; Amrani et al., 2005).

Muscorian (Ling et al., 2007) makes use of a generic POS tagger as part of MontyLingua (Ling, 2006; Liu and Singh, 2004) and performs at a comparable level to IE tools using POS taggers trained on biomedical text. This contradicts the common view that “*error propagation through cascades of processors may in aggregate severely degrade performance on the final task*” as stated in the *Call for Papers for the Tenth Conference on Natural Language Processing 2006 (CoNLL-X)*. Tateisi and Tsujii (2004) have demonstrated that generic POS taggers are only about 83% accurate when used to tag biomedical text. This suggests that MontyTagger, the generic POS tagger in MontyLingua, is unlikely to perform as well as taggers trained on biomedical text, such as MedPost (Smith et al., 2004). Therefore, it is likely that the above mentioned contradiction is resolved at the step immediately downstream to POS tagging, the shallow parsing. In MontyLingua shallow parsing (Ling et al., 2007), the input sentence is broken into noun phrase and verb phrase. The process of shallow parsing can be seen as a collapse of a sequence of POS tags into 2 groups; hence, we expect high level of permissible substitution of POS tags within related classes. We term this permissible substitution as “alternate POS tag use”.

This study compares the performance of MedPost (Smith et al., 2004) with the generic POS tagger, MontyTagger (Liu and Singh, 2004), in Muscorian (Ling et al., 2007) and illustrates a case whereby POS tagging error does not adversely affect the final information extraction task if the errors were resolved in shallow parsing through alternate POS tag use.

## **3.2 Methods**

### **3.2.1 Evaluating POS Tagging and Information Extraction Performance**

MontyTagger was evaluated on its own using MedPost corpus (Smith et al., 2004) and its accuracy as the percentage of the number of correctly tagged tokens (words and punctuations) in the total number of tokens (n=182399). MedPost tagger was swapped

in place of MontyTagger by modifying MontyLingua's *jist()* and *jist\_predicates()* functions to *mpjist()* and *mpjist\_predicates()*, giving MedPost-MontyLingua Muscorian:

```
def jist(self,text):
    sentences = self.split_sentences(text)
    tokenized = map(self.tokenize,sentences)
    tagged = map(self.tag_tokenized,tokenized)
    chunked = map(self.chunk_tagged,tagged)
    extracted = map(self.extract_info,chunked)
    return extracted

def jist_predicates(self,text):
    infos = self.jist(text)
    svoos_list = []
    for info in infos:
        svoos =
        info['verb_arg_structures_concise']
        svoos_list.append(svoos)
    return svoos_list
```

to

```
def mpjist(self,text):
    sentences = self.split_sentences(text)
    tokenized = map(self.tokenize,sentences)
    sourcefilename =
        random.random()*1000000000
    outfilename =
        random.random()*1000000000000
    source = open('temp' + os.sep +
        str(sourcefilename), 'w')
    source.writelines(tokenized)
    source.close()
    os.popen(os.getcwd() + os.sep +
        'medpost/medpost -text -token -penn <
        temp' + os.sep + str(sourcefilename) + '>
        temp' + os.sep + str(outfilename))

    mpout = open('temp' + os.sep +
        str(outfilename), 'r')
    tagged = mpout.readlines()
    mpout.close()
    chunked = map(self.chunk_tagged,tagged)
    extracted = map(self.extract_info,chunked)
    return extracted

def mpjist_predicates(self,text):
    infos = self.mpjist(text)
    svoos_list = []
    for info in infos:
        svoos =
        info['verb_arg_structures_concise']
        svoos_list.append(svoos)
    return svoos_list
```

MedPost-MontyLingua Muscorian's IE performance was evaluated using Learning Languages in Logic 2005 test data (Cussens and Nedellec, 2005) in the same manner as Muscorian (Ling et al., 2007) and the performances were compared (Figure 3a).

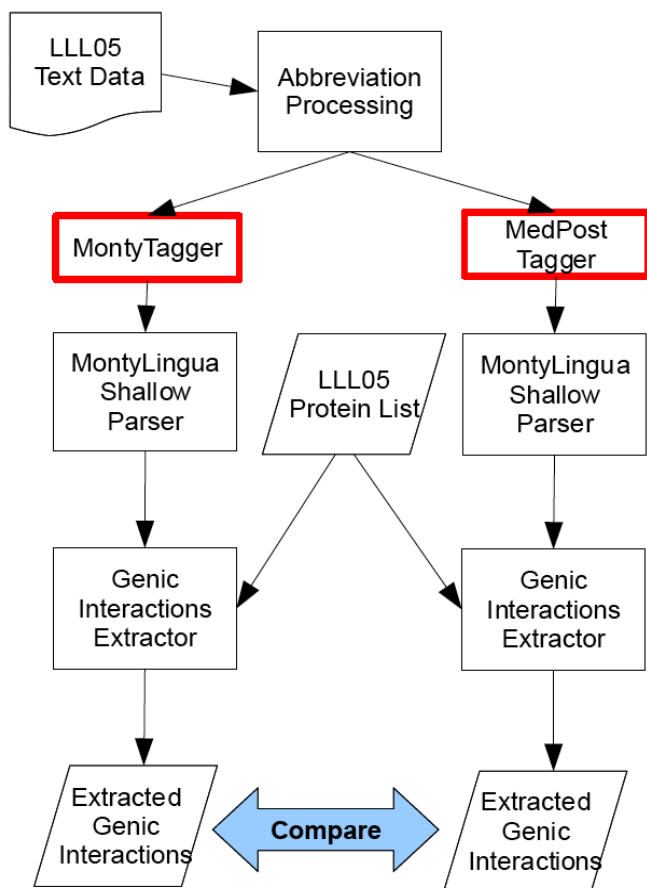


Figure 3a. Flowchart of Evaluation Procedure for Muscorian with Native MontyLingua and MedPost-MontyLingua.  
 LLL05 test data was processed for abbreviations before feeding into each system and the extracted genic interactions (output) were evaluated for precision and recall.

### 3.2.2 Analysis of POS Tagging Errors

Wrongly tagged tokens from MontyTagger's output were first grouped by their original tags in MedPost corpus (Smith et al., 2004), then sub-grouped by MontyTagger's assigned tags (the wrong tag) and arranged in decreasing order based on the numbers of tags in both main and sub-group. The first 90% of each of the wrongly assigned tags were chosen for further error analysis. Each of the pairs of original tag and wrongly assigned tag were analysed with respect to the regular expressions in MontyREChunker (Ling et al., 2007), the shallow parser in MontyLingua, for the effects of the wrongly assigned tags on the operations of the shallow parser.

### 3.3 Results

#### 3.3.1 Evaluating POS Tagging and Information Extraction Performance

Evaluating MontyTagger on MedTag corpus demonstrated correct tagging in 151663 of the tags representing 83.1% tagging accuracy. Using the LLL05 evaluation corpus, Muscorian with MedPost-MontyLingua on directional relationship was found to be 56.8% precise with 24.8% recall, while nondirectional relationship was estimated to be 81.8% precise with 35.6% recall (Table 3a).

Table 3a. Summary of Muscorian's Performances Evaluated Using Learning Languages in Logic 2005 Data (Cussens, 2005).

	<i>Directional Relationships</i>		<i>Nondirectional Relationships</i>	
	MontyLingua	mpMontyLingua	MontyLingua	mpMontyLingua
Precision	55.6%	56.8%	86.1%	81.8%
Recall	19.8%	24.8%	30.7%	35.6%
F-Score	0.292	0.345	0.453	0.496

#### 3.3.2 Analysis of POS Tagging Errors

Comparison of the reference tags (MedPost corpus) with the wrongly assigned tags from MontyTagger showed the 30736 wrongly assigned tags (52.3%, n=16067) should be tagged as nouns (tag: 'NN'), 15.8% (n=4865) should be tagged as 'JJ' (adjectives), and the next four most common wrongly assigned tags were 'NNS' (n=1987, 6.5%), 'SYM' (n=1496, 4.9%), 'VBP' (n=1470, 4.8%), and 'VBD' (n=745, 2.4%). These six reference tags (NN, JJ, NNS, SYM, VBP, VBN) accounted for 26630 (86.6%) of the wrongly assigned tags, while the rest of the errors (n=4106) were distributed across 25 tags. Six tags (TO, :, (, ), WP, ,) were correctly assigned in every instance in this evaluation. A tabulation of errors is shown in Table 3b and a table providing the definition of each POS tag is given in Table 3c. The confusion matrix can be found at [http://ib-dwb.sf.net/Muscorian/MedPost\\_confuse.txt](http://ib-dwb.sf.net/Muscorian/MedPost_confuse.txt).

Table 3b. Percentage Breakdown of POS Tags in MedTag corpus and Errors in MontyTagger as Percentage of POS Tags Assignment. This table tabulates the POS tagging errors made by MontyTagger on MedTag corpus and the order is according to the abundance of each tag in the MedTag corpus. For example, 'NN' is the most abundant tag accounting for 28.56% or 52093 of MedTag corpus of 182399 tokens. Of which, 3084% (16067 of 52093) of the 'NN' tokens in MedTag corpus were wrongly assigned to a different POS tag by MontyTagger which accounted for 52.27% of the total wrongly assigned POS tag of 30736 tokens.

<i><b>Tag</b></i>	<i><b>% Corpus</b></i>	<i><b>% Error in Total Error</b></i>	<i><b>% Error in Tag</b></i>	<i><b>Tag</b></i>	<i><b>% Corpus</b></i>	<i><b>% Error in Total Error</b></i>	<i><b>% Error in Tag</b></i>
NN	28.56	52.27	30.84	VBG	0.64	0.06	1.59
IN	13.49	1.08	1.33	:	0.54	0.00	0.00
JJ	10.47	15.81	25.44	MD	0.43	0.01	0.2
DT	7.77	0.56	1.16	WDT	0.45	0.19	6.70
NNS	7.75	6.45	14.03	,	0.39	0.00	0.00
CC	6.66	1.30	3.29	PRP\$	0.28	0.01	0.40
.	3.67	0.01	0.03	FW	0.26	0.96	61.39
CD	3.13	2.02	10.84	WRB	0.23	0.59	43.33
VCN	3.05	1.70	10.13	JJR	0.17	0.17	17.74
VBD	2.81	2.42	14.56	NNP	0.14	0.03	3.53
RB	2.57	1.72	9.49	EX	0.08	0.01	1.38
)	1.89	0.00	0.00	POS	0.06	0.06	15.31
(	1.88	0.00	0.00	WP	0.06	0.00	0.00
VBP	1.98	4.78	41.26	JJS	0.05	0.02	6.60
TO	1.55	0.00	0.00	RBS	0.05	0.01	4.40
VBZ	1.54	0.45	5.20	“	0.03	0.19	100.00
SYM	1.07	4.87	76.43	”	0.03	0.19	100.00
PRP	0.88	1.61	30.59	PDT	0.02	0.11	100.00
VB	0.74	0.05	1.11	RBR	0.01	0.03	44.44

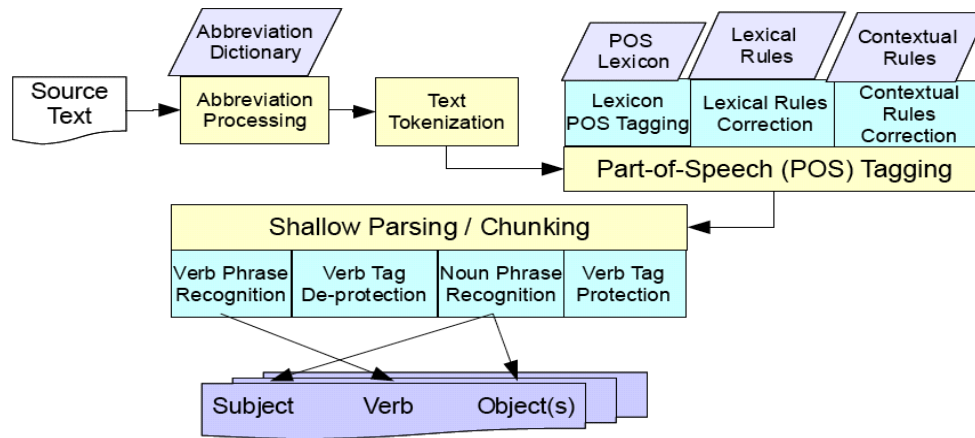


Figure 3b. Muscorian's Generalization Layer, From Source Text to Subject-Verb-Object(s) Structures (Ling et al., 2007).

Table 3c. Penn Treebank Tag Set without Punctuation Tags (Adapted from (Marcus et al., 1993))

Tag	Description	Tag	Description
CC	Coordinating conjunction	PRP\$	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determinant	RBR	Adverb, comparative
EX	Existential <i>there</i>	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	to
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBN	Verb, past participle
NN	Noun, singular or mass	VBG	Verb, gerund or present participle
NNS	Noun, plural	VBP	Verb, non-3 <sup>rd</sup> person singular present
NNP	Proper noun, singular	VBZ	Verb, 3 <sup>rd</sup> person singular present
NNPS	Proper noun, plural	WDT	Wh-determiner
PDT	Predeterminer	WP	Wh-pronoun
POS	Possessive ending	WP\$	Possessive wh-pronoun
PRP	Personal pronoun	WRB	Wh-adverb

An understanding of the general scheme of operations of MontyLingua as described in Ling et al. (2007), especially downstream process of POS tagging, the process of shallow parsing by MontyREChunker (MontyLingua's shallow parser) is crucial in our error analysis (Figure 3b). Source text (abstracts) were processed for abbreviations and tokenized into sentences, then words and punctuations, before POS tagging. In POS tagging, each token was tagged first using a lexicon and corrected using lexical and contextual rules. This was where the output was 83.1% accurate compared to 96.9% in MedPost. POS tagging could be seen as a reduction of potentially unlimited human English words into 45 “words” or tags using knowledge of English grammar, and the sequence of tags was the input to the shallow parser, MontyREChunker. Firstly, verb tags (VBD, VBG and VBN) were protected by suffixing the tags to prevent interference in subsequent noun phrase recognition (Ling et al., 2007). This meant that wrong tagging between these three tags, such as VBD was erroneously tagged as VBN, had no effect on this process. However, wrong tagging of any of these three tags to any of the other 42 tags or the other way around will be detrimental to this process. Secondly, noun phrases were recognized by the following regular expression (according to Python regex engine in the Python standard library):

```
((((PDT )?(DT |PRP[$] |WDT |WP[$] )(VBG |VBD |VBN |JJ |JJR |JJS |, |CC |NN |NNS |NNP |NNPS |CD )*(NN |NNS |NNP |NNPS |CD )+)|((PDT )?(JJ |JJR |JJS |, |CC |NN |NNS |NNP |NNPS |CD )*(NN |NNS |NNP |NNPS |CD )+)|EX |PRP |WP |WDT )POS )?
(((PDT )?(DT |PRP[$] |WDT |WP[$] )(VBG |VBD |VBN |JJ |JJR |JJS |, |CC |NN |NNS |NNP |NNPS |CD )*(NN |NNS |NNP |NNPS |CD )+)|((PDT )?(JJ |JJR |JJS |, |CC |NN |NNS |NNP |NNPS |CD )*(NN |NNS |NNP |NNPS |CD )+)|EX |PRP |WP |WDT )
```

A number of relationships that potentially contribute to reduced POS tagging errors were considered. Firstly, these four tags; DT, PRP[\$], WDT, and WP[\$]; were alternatives to each other and erroneous tagging between them had no effect on shallow parsing. Secondly, the ten tags; JJ, JJR, JJS, “,” ,CC, NN, NNS, NNP, NNPS, and CD; were alternatives to each other. Lastly, these four tags; EX, PRP, WP, and WDT; were alternatives to each other.

Subsequently, verb phrases were de-protected by removing the suffix appended during

the tag protection phase (Ling et al., 2007), followed by verb phrase recognition. This meant that verb tag protection had the highest precedence, followed by noun phrase recognition, and then verb phrase recognition. This meant that nullified errors in higher precedence would not affect downstream processes. Verb phrases were recognized by the following regular expression:

`(RB |RBR |RBS |WRB )*(MD )?(RB |RBR |RBS |WRB )*(VB |VBD |VBG |VBN |VBP |VBZ )(VB |VBD |VBG |VBN |VBP |VBZ |RB |RBR |RBS |WRB )*(RP )?(TO (RB )*(VB |VBN )(RP )?)?`

In terms of compensation for POS tagging errors, this meant that the four tags; RB, RBR, RBS, and WRB; and these six tags; VB, VBD, VBG, VBN, VBP, and VBZ; were alternatives to each other. However, verb phrase required terminal VB or VBN, which meant that although verb tag protection allowed for interchangeable use of VBN, VBG and VBD, erroneous tagging of VBG and VBD to VBN or VBN to VBG or VBD would be detrimental to verb phrase recognition.

A deeper analysis was undertaken to examine the errors in each reference tag (tabulated in Table 3d). Firstly, by grouping close POS types, for example 'NN', 'NNP', and 'NNS' were all nouns, wrong sub-type assignation, such as 'NN' assigned as 'NNP' and 'NNS' assigned as 'NNP', accounted for 55% of the errors (n=14634). Secondly, 58% (n=2818) of 'JJ' (adjective) errors were resulted by tagging as noun (NN and NNP) while 34.9% (n=1698) of the 'JJ' errors were tagged as verb (VBN and VBG). Thirdly, about 5% (n=941) of 'NN' (noun) errors were tagged as cardinal numbers (CD). Fourthly, plural nouns accounted for 51.6% (n=1026) of 'NNS' (singular noun) errors. Fifthly, 48.7% (n=729) and 39.3% (n=587) of 'SYM' (symbol) errors were either not assigned or assigned as 'NN' (noun) respectively. Lastly, 87% (n=1927) of verb errors (VBP and VBD) were due resolution of tenses, such as non-third party singular present tense (VBP) was assigned as infinite verb form (VB).

Error breakdown (in Table 3d) demonstrated erroneous POS tagging by MontyTagger in 31 tags, with 6 tags having no errors. A total of 6 of the 32 tags (19.4%) accounted for 86.6% (n=26630) of the total errors and were chosen for further analysis. Applying these error nullification rules to each of the examined erroneous tags (86.6% of the



errors), it was found that 78.6% of the errors had no effect on shallow parsing. A tabulated analysis is shown in Table 3d.

Table 3d. Error Breakdown and Analysis on the Effects of Six Most Commonly Mis-Assigned POS Tags. Six reference tags; NN, JJ, NNS, SYM, VBP, and VBD; which accounted for 86.6% of all wrong POS assignation by MontyTagger were chosen and in each tag, the assigned tags which accounted for 90% of the errors were chosen for further analysis. For example, of 16067 tags that were tagged as 'NN' in MedTag corpus, MontyTagger wrongly tagged 10865 tokens as 'NNP' and has no effect on shallow parsing, 2527 tokens as 'JJ' and has no effect on shallow parsing, 941 tokens as 'CD' and has no effect on shallow parsing, and 812 tokens as 'VBG' with an effect on shallow parsing. These 4 wrong tagging accounted for 94.3% of all 'NN' tag errors. This also meant that 922 'NN' tag errors (5.7%) were not further analyzed. A complete confusion matrix is given in <http://ib-dwb.sf.net/Muscorian/MedPost-confuse.txt>.

<i>Reference Tag</i>	<i>Wrongly Assigned Tag</i>	<i>Number of Wrong Assignment</i>	<i>Cummulative Frequency for Reference Tag</i>	<i>Impact on Shallow Parsing?</i>
NN (16067)	NNP	10865	67.6%	No, NNP was an alternative match to NN in noun phrase recognition
	JJ	2527	83.4%	No, JJ was an alternative match to NN in noun phrase recognition
	CD	941	89.2%	No, CD was an alternative match to NN in noun phrase recognition
	VBG	812	94.3%	Yes, protected verb tag
JJ (4865)	NN	1600	32.9%	No, NN was an alternative match to JJ in noun phrase recognition
	NNP	1218	58.0%	No, NNP was an alternative match to JJ in noun phrase recognition
	VBN	1170	82.0%	Yes, protected verb tag
	VBG	528	92.8%	Yes, protected verb tag
NNS (1987)	NNP	1026	51.6%	No, NNP was an alternative match to NNS in noun phrase recognition
	NN	701	86.9%	No, NN was an alternative match to NNS in noun phrase recognition
	VBZ	128	93.4%	No, VBZ was an alternative match to NNS in noun

<i>Reference Tag</i>	<i>Wrongly Assigned Tag</i>	<i>Number of Wrong Assignment</i>	<i>Cummulative Frequency for Reference Tag</i>	<i>Impact on Shallow Parsing?</i>
				phrase and was not a protected verb tag
SYM (1496)	Not Assigned	729	48.7%	No, tokens not tagged were non-existent and SYM was not used in shallow parsing
	NN	587	88.0%	Yes, NN was matched in noun phrase
	-	115	95.7%	No, both tags was not used in shallow parsing
VBP (1470)	VB	1249	85.0%	Yes, mandatory requirement of VB in verb phrase
	NN	178	97.1%	No, NN was an alternative match to VBP in noun phrase
VBD (745)	VBN	678	91.0%	Yes, mandatory requirement of VBN in verb phrase
	JJ	34	95.6%	Yes, protected verb tag

### 3.4 Discussion

The precision and recall of native MontyLingua Muscorian for extracting genic interactions from the LLL05 data set (Cussens and Nedellec, 2005) was 55.6% and 19.7% (F-score = 0.29) respectively for directional interactions which is about 5% higher in precision and similar in recall to that reported in LLL05 (Cussens and Nedellec, 2005). The precision and recall was 86.1% and 30.7% (F-score = 0.45) respectively for nondirectional interaction. The term “directional” means that the direction of protein activity is non-commutative, for example, “proteinA activates proteinB” does not the same as “proteinB activates proteinA”. However, nondirectional means that the protein activity is commutative, for example, “proteinA binds to proteinB” has no different biological significance than “proteinB binds to proteinA”. This formed the baseline to evaluate a biomedical-specialized part-of-speech (POS) tagger (Smith et al., 2004) modification of Muscorian, MedPost-MontyLingua Muscorian. The main reason for examining this specialized POS tagger was that it was developed for biomedical information extraction systems (Daniel et al., 2004; Chen and

Sharp, 2004) and POS tagging errors were known to be detrimental in understanding human text (Kodratoff et al., 2005; Amrani et al., 2005). In addition, POS tagger modification had been done in a number of biomedical information extraction systems, such as Jang et al. (2006) and Chilobot (Chen and Sharp, 2004).

Examining MontyLingua's source codes, the main function that processes text is the *jist\_predicate()* function, which calls the *jist()* function to process text (tokenization, POS tagging and shallow parsing) and then to extract the resulting set of subject-verb-objects (SVO) from *jist*'s output (Ling et al., 2007). The Python codes for these two functions were as follows:

```
def jist(self,text):
    sentences = self.split_sentences(text)
    tokenized = map(self.tokenize,sentences)
    tagged = map(self.tag_tokenized,tokenized)
    chunked = map(self.chunk_tagged,tagged)
    extracted = map(self.extract_info,chunked)
    return extracted

def jist_predicates(self,text):
    infos = self.jist(text)
    svoos_list = []
    for info in infos:
        svoos =
        info['verb_arg_structures_concise']
        svoos_list.append(svoos)
    return svoos_list
```

As observed, *jist()* function calls *tokenize* function to tokenize the text, *tag\_tokenized* function to perform POS tagging, *chunk\_tagged* function to perform shallow parsing, and finally, *extract\_info* function to extract SVOs from the parsed text. The systematic structure of MontyLingua's codes, especially the *jist()* function had simplified the substitution of MontyTagger (by *tag\_tokenized* function) with MedPost. This implied that any of the other components in the text analysis process, like shallow parser (by *chunk\_tagged* function) could be easily exchanged.

The precision and recall of MedPost-MontyLingua Muscorian evaluated using the LLL05 data set (Cussens and Nedellec, 2005) were 56.8% and 24.8% (F-score = 0.35) respectively for directional interactions, and 81.8% and 35.6% (F-score = 0.50) respectively for nondirectional interaction. Our results showed that using MedPost in place of MontyLingua's POS tagger, MontyTagger, had improved the F-score by about 5% in both directional and nondirectional interactions extraction, and recall (24.8% versus 19.7% and 35.6% versus 30.7%). However, as reasoned in Ling et al. (2007), precision was more important than recall when extracted protein-protein interactions were used to support other biological analyses and the problem with mediocre recall is resolved with large volumes of text.

Our results indicated that MedPost-MontyLingua Muscorian outperformed un-modified-MontyLingua Muscorian in extracting directional genic interactions in terms of both precision and recall, suggesting that MedPost-MontyLingua Muscorian was more suited for this purpose. However, the precision of MedPost-MontyLingua Muscorian underperformed in extracting non-directional genic interactions, despite better recall. This suggested that errors in MontyTagger (un-modified-MontyLingua's POS tagger) resulted in more directional errors than that of MedPost. Given that our interest was in nondirectional interactions and precision was more important than recall in our case, un-modified-MontyLingua Muscorian was chosen for future work.

We conclude that our experimental results indicated that un-modified-MontyLingua Muscorian performed as well as MedPost-MontyLingua Muscorian for the purpose of processing biomedical text for the extraction of genic interactions. Thus, in contrary to the general assumption that generic text processing systems must be modified before being suitable for processing biological text for extracting genic interactions as evident from numerous systems to date, we presented a case study where comparable performance could be achieved by using generic text processing tools. This outcome is consistent with a previous study using un-modified MontyLingua for processing peer-reviewed economics papers (van Eck, 2005; van Eck and van Den Berg, 2005).

An initial evaluation of MontyTagger on MedTag Corpus (Smith et al., 2004) indicated 83.1% accuracy, which was considerably less than from MedPost's reported accuracy of 96.9% (Smith et al., 2004) and was close to the 83.0% tagging accuracy of a generic POS tagger on biomedical text (Tateisi and Tsujii, 2004). This result was expected as MontyTagger was not developed for biomedical text (Ling et al., 2007).

The POS tagging errors were expected to impact on performance of the entire text processing pipeline but this was not observed in our results. Instead, the precision of un-modified-MontyLingua Muscorian was comparable to that of MedPost-MontyLingua Muscorian on directional genic interactions (55.6% versus 56.8%) and un-modified-MontyLingua Muscorian outperformed MedPost-MontyLingua Muscorian on nondirectional genic interactions (86.1% versus 81.8%). Taken collectively the precision of both system and their respective POS tagging accuracies, seemed

contradictory to general expectations as stated in the *Call for Papers for the Tenth Conference on Natural Language Processing 2006 (CoNLL-X)*.

An error analysis on MontyTagger was carried out in attempt to provide insight into resolving this contradiction. A likely hypothesis to explain why POS tagging errors did not derail the entire text processing pipeline was that the errors were nullified post-tagging. Text processing is used in Muscorian as a means to convert unstructured text into structured form for data mining - an extremely limited use of natural language processing compared to more complex uses, such as automated translation. As mentioned previously, POS tagging can be seen as a process of mapping potentially infinite number of words in the English language into a finite set of tags, based on their syntactic meanings. Shallow parsing, also known as chunking, can then be seen as a process which examines the sequence of tags and splits them into semantic phrases, of which verb phrase and noun phrase are of interest in this case. Given that MontyLingua's shallow parser parses the sequence of tags into 3 types of phrases (verb, noun, and adjectives), it is conceivable that a number of POS errors have no effect on shallow parsing.

Of the 182399 token in MedTag Corpus (Smith et al., 2004), 30736 were erroneously tagged by MontyTagger (16.9% error) spreading over 40 tags. The top 6 most common tag errors accounted for 86.6% of the total errors and were chosen for further evaluation. In each of the 6 most abundant error tags, the top 95% of the errors were examined.

The effects of each type of errors, such as 'NN' wrongly tagged to 'NNP', were examined by analyzing the routines for shallow parsing which uses Regular Expressions. It was found that in 26630 of the examined POS tagging errors, 20928 (78.6% of 26630) had no effect on the chunking process and the remaining 5703 errors adversely affected shallow parsing, which might account for lower recall of unmodified-MontyLingua Muscorian as compared to MedPost-MontyLingua Muscorian.

Therefore, despite a low POS tagging accuracy of 83.1% by MontyTagger, more than three-quarters of the errors had no detrimental effect on chunking, suggesting a

“functional POS tagging accuracy” of at least 94.6%, which was relatively close to MedPost's reported 97% accuracy (Smith et al., 2004). This apparent high “functional POS tagging performance” despite poor actual tagging accuracy might be the reason to explain un-modified-MontyLingua Muscorian's good performance in LLL05 test (Cussens and Nedellec, 2005) despite poor tagging accuracy compared to MedPost-MontyLingua Muscorian. This suggested that the nature of POS tagging errors might be more important than a single measure of POS tagging accuracy in a specific use of generic text processing tools where a shallow parser is involved. Therefore, it can be inferred that applications of biomedical literature analysis where a shallow parser is likely to be involved, such as extracting entity interactions and protein or molecule localization, POS tagging errors may not result in a decline in system performance.

At the same time, it is known that building domain-specific text processing tools requires much manual efforts (Jensen et al., 2006) suggesting that the cost and effort needed to train taggers specifically for biomedical text may not be needed, depending on the target application. However, it should also be cautioned that other applications or systems that do not involve shallow parser, such as Arizona Relation Parser (Daniel et al., 2004) which uses full sentence parsing, are likely to benefit from superior POS tagging accuracy of MedPost (Smith et al., 2004) and may experience degraded results from tagging errors.

MedTag Corpus (Smith et al., 2004) was used as a standard for evaluating MontyTagger. However, only 38 of the 45 tags in Penn Treebank Tag Set were used to annotate the corpus while the tagged output of MontyTagger illustrated the use of 45 tags. This might suggest inconsistencies or errors in MedPost Corpus, which were found in other POS tagged corpora (Peshkin and Savova, 2003; Ratnaparkhi, 1996).

### **3.5 Conclusions**

In summary, analysis of the effects of MontyTagger's errors on downstream shallow parsing by MontyREChunker illustrated that 78.6% of the examined errors had no effect on shallow parsing. This implied that although the POS tagging accuracy of MontyTagger on MedPost Corpus was 83.1%, a majority of the errors had no downstream effect; thus, the functional POS tagging accuracy of MontyTagger was

between 94.6% and 96.9%. A good functional POS tagging accuracy despite poor POS tagging accuracy, with respect to shallow parsing, is a likely reason for a comparative performance in extracting protein-protein interactions from text using a domain-specific or a generic POS tagger.

## Chapter 4

# Filtering Microarray Correlations by Statistical Literature Analysis Yields Potential Hypotheses for Lactation Research

### 4.1 Introduction

Microarray technology is a transcriptome analysis tool which had been used in the study of the mouse lactation cycle (Clarkson and Watson, 2003; Rudolph et al., 2007). A number of advances in microarray analysis have been made recently. For example, inferring the underlying genetic network from microarray results (Rawool and Venkatesh, 2007; Maraziotis et al., 2007) by statistical correlation of gene expression across a series of samples (Reverter et al., 2005), then deriving functional network clusters by mapping onto Gene Ontology (Beissbarth, 2006). It has been shown that functionally related genes demonstrate similar expression profiles (Reverter et al., 2005). These methods have been used to study functional gene sets for basal cell carcinoma (O'Driscoll et al., 2006). The amount of information in published form is increasing exponentially, making it difficult for researchers to keep abreast with the relevant literature (Hunter and Cohen, 2006). At the same time, there has been no study to demonstrate that the current status of knowledge in protein-protein interactions in the literature is useful to increase the understanding of microarray data.

The two major streams for biomedical protein-protein information extraction are natural language processing (NLP) and co-occurrence statistics (Cohen and Hersh, 2005; Jensen et al., 2006). Co-occurrence is statistical method based on the tenet that multiple occurrences of the same pair of entities suggests that the pair of entities are related in some way (Cohen and Hunter, 2007) whereas NLP uses the grammatical structure of the underlying language (English in the this case) to deduce the relationships between two entities. The main reason for concurrent existence of these two methods is their complementary effect in terms of information extraction (Jensen et al., 2006). NLP has a



lower recall or sensitivity than co-occurrence but tends to be more precise compared with co-occurrence statistical methods (Wren and Garner, 2004; Jensen et al., 2006). Mathematically, precision is the number of true positives divided by the total number of items labeled by the system as positive (number of true positives divided by the sum of true and false positives), whereas recall is the number of true positives identified by the system divided by the number of actual positives (number of true positives divided by the sum of true positives and false negatives). A number of tools have approached protein-protein interaction extraction from the NLP perspective, these include GENIES (Friedman et al., 2001), MedScan (Novichkova et al., 2003), PreBIND (Donaldson et al., 2003), BioRAT (David et al., 2004), GIS (Chiang et al., 2004), CONAN (Malik et al., 2006), and Muscorian (Ling et al., 2007). Muscorian (Ling et al., 2007) achieved at least 82% precision and 30% in recall (sensitivity). NLP methods made use of the grammatical forms of words and structure of a valid sentence to identify the grammatical roles of each word in a sentence, parse the sentence into phrases and extracting information such as subject-verb-object structures from these phrases. Co-occurrence, a statistical method, is based on the thesis that multiple occurrences of the same pair of entities suggests that the pair of entities are related in some way and the likelihood of such relatedness increases with higher co-occurrence. In other words, co-occurrence methods tend to view the text as a bag of un-sequenced words. Hence, depending on the threshold allowed, which will translate to the precision of the entire system, recall could be total, as implied in PubGene (Jenssen et al., 2001).

PubGene (Jenssen et al., 2001) defined interactions by co-occurrence to the simplest and widest possible form by assigning an interaction between 2 proteins if these 2 proteins appear in the same article just once in the entire library of 10 million articles and found that this criterion has 60% precision (1-Mention PubGene method). Although it was not stated in the article (Jenssen et al., 2001), it is obvious that such a criterion would yield 100% recall or sensitivity, giving an F-score of 0.75. F-score is defined as the harmonic mean of precision and recall, attributing equal weight to both precision and recall. However, 60% precision is usually unsatisfactory for most applications. PubGene (Jenssen et al., 2001) had also defined a “5-Mention” method which requires 5 or more articles with 2 protein names to assign an interaction with 72% precision. It is generally accepted that precision and recall are inversely related; hence, it can be expected that

the “5-Mention” method will not be 100% sensitive. However, PubGene was benchmarked against the Database of Interacting Proteins and OMIM, making it more difficult to appreciate the statistical basis of “1-Mention” and “5-Mention” methods as compared to using a hypothesis testing framework in Chen et al. (2008). In addition, PubGene is unable to extract the nature of interactions, for example, binding or inhibiting interactions. On the other hand, NLP is designed to extract the nature of interactions (Malik et al., 2006; Ling et al., 2007); hence, it can be expected that NLP results may be used to annotate co-occurrence results.

CoPub Mapper used a more sophisticated information measure which took into account the distribution of entity names in the text database (Alako et al., 2005). Although Alako et al (2005) demonstrated CoPub Mapper's information measure co-relates well with microarray co-expression, the information measure was not used as a decision criterion for deciding which pairs of co-occurrences were positive results (personal communication, Guido Jenster, 2006). This is unlike 1-Mention PubGene method where all co-occurrence were taken as positive result and 5-Mention PubGene method requires at least 5 count of co-occurrence before attributing the co-occurrence as a positive result. Chen et al. (2008) used chi-square to test co-occurrence statistically to mine disease-drug interactions from clinical notes and published literature. Another possible way to calculate co-occurrence is a direct use of Poisson distribution on the assumption that co-occurrence of 2 protein names is a rare chance with respect to the entire library. Poisson distribution is a discrete distribution similar to Binomial distribution but is used for rare events, for example, to estimate the probability of accidents in a given stretch of road in a day. Poisson distribution is easier to use than Binomial distribution as it only requires the mean and does not require a standard deviation. Based on PubGene, the statistical assumption of Poisson distribution-based statistics requiring rare events (in this case, the co-occurrences of 2 protein names in a collection of text is statistically rare) can generally be held (Jenssen et al., 2001).

Although a combination of either NLP or co-occurrence in microarray analysis has been used (Li et al., 2007; Gajendran et al., 2007; Hsu et al., 2007), neither method had been used in microarray analysis for advancing lactational biology. This study attempts to examine the relation between the PubGene and Poisson distribution methods of

calculating co-occurrence and explore the use of NLP-based protein-protein interaction extraction results to annotate co-occurrence results. This study also examines the use of co-occurrence analysis on 4 publically available microarray data sets on mouse lactation cycle (Master et al., 2002; Clarkson and Watson, 2003; Stein et al., 2004; Rudolph et al., 2007) as a novel hypothesis discovery tool. Master et al. (2002) used 13 microarrays to discover the presence of brown adipose tissue in mouse mammary fat pad and its role in thermoregulation. Clarkson and Watson (2003) used 24 microarrays and characterized inflammation response genes during involution. Stein et al. (2004) used 51 microarrays and discovered a set of 145 genes that are up-regulated in early involution where 49 encoded for immunoglobulins. Rudolph et al. (2007) used 29 microarrays to study lipid synthesis in the mouse mammary gland following diets of various fat content and found that genes encoding for nutrient transporter into the cell are up-regulated following increased food intake. More importantly, each of the 4 studies independently demonstrated that the cyclical nature of mammary gland development, as observed histologically and biochemically, are reflected at the transcriptome level suggesting that microarray is a suitable tool to study the regulation of mouse lactation. It should be noted that even-though each of these microarray experiments were designed for different purposes, the principle that co-expressed genes are more functionally correlated than functionally unrelated genes remains, as demonstrated by Reverter et al. (2005).

Our results demonstrate that 5-mention PubGene method is generally statistically more significant than 99<sup>th</sup> percentile of Poisson distribution method of calculating co-occurrence. Our results showed that 96% of the interactions extracted by NLP methods (Ling et al., 2007) overlapped with the results from 5-mention PubGene method. However, less than 2% of the microarray correlations were found in the co-occurrence graph extracted by 1-mention PubGene method. Using co-occurrence results to filter microarray co-expression correlations, we have discovered a potentially novel set of 7 protein-protein interactions that had not been previously described in the literature.

## 4.2 Methods

### 4.2.1 Microarray Datasets

The 4 microarray datasets are from Master et al. (2002) using Affymetrix Mouse Chip Mu6500 and FVB mice (<http://www.abramsoninstitute.org/chodosdata.html>), Clarkson and Watson (2003) using Affymetrix U74Av2 chip and C57/BL6 mice (<http://www.path.cam.ac.uk/~madgroup/Microarray> Data.xls), Rudolph et al. (2007) using Affymetrix U74Av2 chip and FVB mice (GEO accession: GSE4222), and Stein et al. (2004) using Affymetrix U74Av2 chip and Balb/C mice (<http://breast-cancer-research.com/content/supplementary/bcr753-S1.txt>).

### 4.2.2 Co-Occurrence Calculations

Using a pre-defined list of 3653 protein names which was derived by Ling et al. (2007) from Affymetrix Mouse Chip Mu6500 microarray probeset, PubGene established 2 measures of binary co-occurrence (Jenssen et al., 2001): 1-mention method and 5 mentions method. In the 1-mention method, the appearance of 2 entity names in the same abstract will be deemed as a positive outcome whereas the 5 mentions method will require the appearance of 2 entity names in at least 5 abstracts before considered positive. Thus, both 1-mention and 5-mention PubGene methods are count-based methods.

For co-occurrence modelled on Poisson distribution (Poisson co-occurrence), the number of abstracts in which both entity names appeared in is assumed to be rare as it only requires the appearance of 2 entity names within 5 articles in a collection of 10 million articles to give a precision of 0.72 (Jenssen et al., 2001). The relative occurrence frequencies of each of the 2 entities were calculated separately as a quotient of the number of abstracts in which an entity name appeared in and the total number of abstracts in the corpus. The product of relative occurrence frequency of each of the 2 entities can be taken as the mean expected probability of the 2 entities appearing in the same abstract if they are not related, which when multiplied by the total number of abstracts, can be taken as the mean number of occurrence ( $\lambda$ ) of Poisson distribution. For example, if proteinA and proteinB are found in 1000 abstracts each and there are 1 million abstracts, the relative occurrence frequency will be 0.001 each and

the mean number of occurrences will be 1 ( $0.001^2 \times 1000000$ ). This means that we expect 1 abstract in a collection of 1 million to contain proteinA and proteinB if they are not related ( $n = 1, p = 0.5$ ).

A positive result is where the number of abstracts in which both the 2 entities in question appeared on or above the 95<sup>th</sup> (one-tail  $P < 0.05$ ) or 99<sup>th</sup> (one-tail  $P < 0.01$ ) percentile of the Poisson distribution. In both co-occurrence calculations, entity (protein) names in text is recognized by pattern matching, as used in Ling et al. (2007).

### 4.2.3 Comparing Co-Occurrence and Text Processing

Two sets of comparisons were performed: within the different forms of co-occurrence, and between co-occurrence and text processing methods. The first set of comparison aims to evaluate the differences between the 3 co-occurrence methods described above. PubGene's 1-mention and 5-mentions methods were co-related singly and in combination with Poisson co-occurrence methods.

Given that the nodes (N) of a co-occurrence network represents the entities and the links or edges (E) between each node to represent a co-occurrence under the method used, the entire co-occurrence graph ( $G = \{N, E\}$ ), that is, a set of nodes and a set of edges. In addition, given that the same set of entities was used (same set of nodes), the differences between the 2 graphs resulting from 2 co-occurrence methods can then be simply denoted as the number of differences between the 2 sets of edges (subtraction of one set of edges with another set of edges). In practice, a total space model is used. A graph of total possible co-occurrence is where each node is “linked” or co-occurred with every node, including loops (edge to itself). Thus, a graph of total possible co-occurrence has 3653 nodes and 12694969 ( $3563^2$ ) edges. We define a graph,  $G^*$ , as the undirected graph of total possible co-occurrence without parallel edges including loops.  $G^*$  has 3653 nodes and 63457030 [ $3563 \times (3563 - 1) / 2$ ] edges. The output graph of each co-occurrence method is reduced to the number of edges it contains as it can be assumed that the graph from 1-mention PubGene method represents the most liberal co-occurrence graph ( $G_{PG1}$ ), the resulting graph from any other more sophisticated method ( $G_i$  where  $i$  denotes the co-occurrence method) will be a proper subset of  $G_{PG1}$  and certainly  $G^*$ .

The second set of comparison aims at correlating co-occurrence techniques and natural language processing techniques for extracting interactions between two entities, such as two proteins. In this comparison, the extracted protein-protein binding and activation interactions, extracted using Muscorian on 860000 published abstracts using “mouse” as the keyword as previously described (Ling et al., 2007), has been used to compare against the co-occurrence network of 1-Mention PubGene and 5-Mention PubGene by graph edges overlapping as described above. Briefly, Muscorian (Ling et al., 2007) normalized protein names within abstracts by converting the names into abbreviations before processing the abbreviated abstracts into a table of subject-verb-objects. Protein-protein interaction extractions were carried out by matching of each of the 12694969 (3563<sup>2</sup>) pairs of protein names and verb, namely, activate or bind, in the extracted table of subject-verb-objects.

#### **4.2.4 Mapping Co-Expression Networks onto Text-Mined Networks**

A co-expression network was generated from each of the 4 in vivo data sets by pairwise calculation of Pearson's coefficient on the intensity values across the dataset, where a coefficient of more than 0.75 or less than -0.75 signifies the presence of a co-expression between the pair of signals on the microarray (Reverter et al., 2005). The co-expression network generated from Master et al. (2002) and an intersected co-expression network generated by intersecting all 4 networks were used to map onto 1-PubGene and NLP-mined networks. The co-expression network generated from Master et al. (2002) was mapped against each of the co-occurrence networks. A co-occurrence network was generated from correlation coefficient of 0.75 to 1, with a 0.01 coefficient unit increments, yielding a total of 25 co-occurrence networks.

### **4.3 Results**

#### **4.3.1 Comparing Co-Occurrence Calculation Methods**

Using 3563 transcript names, there is a total of 6345703 possible pairs of interactions - 927648 (14.6%) were found using 1-Mention PubGene method and 431173 (6.80%) were found using 5-Mention PubGene method. The Poisson co-occurrence method

using both 95<sup>th</sup> or 99<sup>th</sup> percentile threshold found 927648 co-occurrences, which is the same set as using 1-Mention PubGene method.

The mean number of co-occurrences, which is used as the mean of the Poisson distribution, is calculated as the product of the probability of occurrence of each of the entity names in the database. Using a database of 100 thousand abstracts as an example, if 500 abstracts contained the term “insulin” (500 abstracts in 100 thousand, or 0.5%) and 200 abstracts contained the term “MAP kinase” (200 abstracts in 100 thousand, or 0.2%), then the mean number of co-occurrence (lambda in Poisson distribution) is 0.001%. The range of mean number of co-occurrences for the 6345703 pairs of entities were from zero to 0.59, with mean of 0.000031. For example, if the mean is  $3.1 \times 10^{-5}$ , then the probability of an abstract mentioning 2 proteins not related in any functional way is  $4.8 \times 10^{-10}$  or virtually zero in 6.3 million possible interactions. These results are summarized in Table 4a.

Table 4a. Summary Results of Co-Occurrence Using PubGene or Poisson Distribution

	<b>Number of Clone-Pairs</b>	<b>% of Full Combination</b>
Full Combination (G*) <sup>1</sup>	6345703	100.00
1-Mention PubGene	927648	14.62
5-Mention PubGene	431173	6.80
Poisson Co-occurrence at 95 <sup>th</sup> percentile	927648 <sup>2</sup>	14.62
Poisson Co-occurrence at 99 <sup>th</sup> percentile	927648 <sup>2</sup>	14.62

<sup>1</sup> The undirected graph of total possible co-occurrence (35632) without parallel edges excluding self edge, which has 3653 nodes and 63457030 [ $3563 \times (3563 - 1) / 2$ ] edges.

<sup>2</sup> Same set as 1-Mention PubGene

### 4.3.2 Comparison of Natural Language Processing and Co-Occurrence

Natural language processing (NLP) techniques were used to extract protein-protein binding interactions and protein-protein activation interactions from almost 860000

abstracts as described in Ling et al. (2007). A total of 9803 unique binding interactions and 11365 unique activation interactions were identified, of which 2958 were both binding and activation interactions. Of the 9803 binding interactions, 9661 interactions concurred with 1-Mention PubGene method (98.55%) and 9465 interactions with 5-Mention PubGene method (96.54%). Of the 11365 activation interactions, 11280 interactions and 11111 interactions concurred with 1-Mention PubGene method (99.25%) and 5-Mention PubGene method (97.77%) respectively. Hence, of the 927648 interactions found using 1-Mention PubGene method, 1.04% ( $n = 9661$ ) were binding interactions and 1.22% ( $n = 11280$ ) were activation interactions. Furthermore, of the 431173 interactions found using 5-Mention PubGene method, 2.20% ( $n = 9465$ ) of the interactions were binding interactions and 2.58% ( $n = 11111$ ) were activation interactions. Combining binding and activation interactions ( $n = 18120$ ), 1.96% of 1-Mention PubGene co-occurrence graph and 3.85% of 5-Mention PubGene co-occurrence graph were annotated respectively.

### **4.3.3 Mapping Co-Expression Networks onto Text-Mined Networks**

Using Pearson's correlation coefficient to signify the presence of a co-expression between the pair of spots (genes) on the Master et al. (2002) data set, there are 210283 correlations between -1.00 to -0.75 and 0.75 to 1.00, of which 2014 (0.96% of correlations) are found in 1-PubGene co-occurrence network, 342 (0.16% of correlations) are found in activation network extracted by natural language processing means and 407 (0.19% of correlations) are found in binding network extracted by natural language processing means.

From incremental correlation mapping with 1-PubGene network (tabulated in Table 4b and graphed in Figure 4a), there is a decline of the number of correlations from 208269 (correlation coefficient of 0.75) to 7 (correlation coefficient of 1.00). The percentage of overlap between co-occurrence and co-expression rose linearly from correlation coefficient of 0.75 to 0.85 ( $r = 0.959$ ) while that of correlation coefficient of 0.86 to 0.92 is less correlated ( $r = 0.223$ ). The 7 pairs of correlations in Master et al. (2002) data set with correlation coefficient of 1.00 are; lactotransferrin (Mm.282359) and solute



carrier family 3 (activators of dibasic and neutral amino acid transport), member 2 (Mm.4114); B-cell translocation gene 3 (Mm.2823) and UDP-Gal:betaGlcNAc beta 1,4-galactosyltransferase, polypeptide 1 (Mm.15622); gamma-glutamyltransferase 1 (Mm.4559) and programmed cell death 4 (Mm.1605); FK506 binding protein 11 (Mm.30729) and signal recognition particle 9 (Mm.303071); FK506 binding protein 11 (Mm.30729) and Ras-related protein Rab-18 (Mm.132802); casein gamma (Mm.4908) and casein alpha (Mm.295878); G protein-coupled receptor 83 (Mm.4672) and recombination activating gene 1 activating protein 1 (Mm.17958). The amount of overlap between microarray correlations and 1-mention PubGene co-occurrence increased steadily from 0.96% at the correlation coefficient of 0.75 to 1.057% at the correlation coefficient of 0.87.

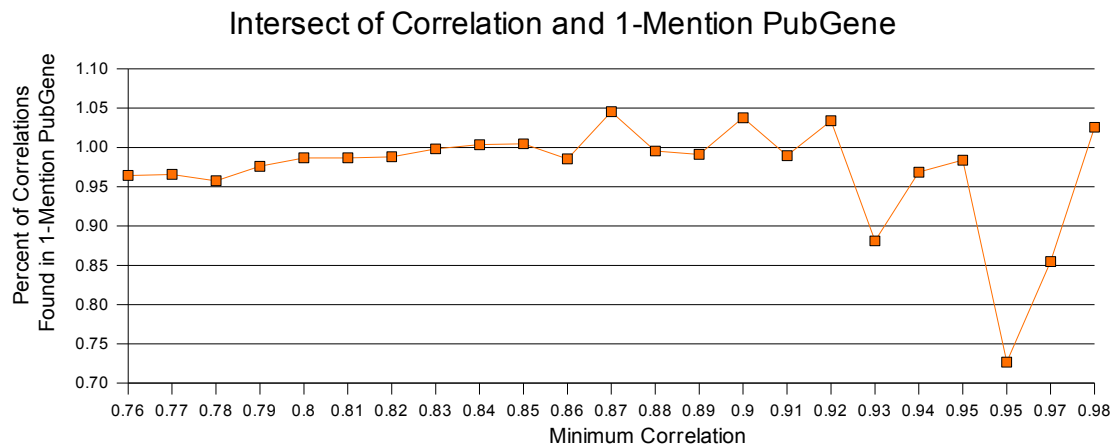


Figure 4a. Percentage of Correlation Network Analyzed from Master et al. (2002) are Found in 1-Mention PubGene Co-Occurrence.

Mapping an intersect of co-expression networks of all 4 in vivo data sets (Master et al., 2002; Clarkson and Watson, 2003; Stein et al., 2004; Rudolph et al., 2007), there are 1140 correlations, of which 14 (1.23%) are found in 1-PubGene co-occurrence network, none of which corresponds to the interactions found in activation or binding networks extracted by natural language processing means (Ling et al., 2007).

Table 4b. Summary of Incremental Stepwise Mapping of Correlation Coefficients from Master et al. (2002) to 1-PubGene Co-Occurrence Network.

<b>Minimum Correlation</b>	<b>Number of Correlations in Master et al. (2002)</b>	<b>Number of Correlations found in 1-PubGene</b>	<b>Percentage of Correlations Found</b>
0.75	210283	2014	0.958
0.76	207593	1983	0.964
0.77	181383	1735	0.966
0.78	157622	1495	0.958
0.79	136152	1316	0.976
0.80	116775	1141	0.987
0.81	99276	970	0.987
0.82	83802	823	0.988
0.83	70019	692	0.998
0.84	57872	575	1.004
0.85	47453	472	1.005
0.86	38228	373	0.985
0.87	30347	314	1.046
0.88	23740	234	0.995
0.89	18137	178	0.991
0.90	13435	138	1.038
0.91	9797	96	0.990
0.92	6849	70	1.034
0.93	4580	40	0.881
0.94	2919	28	0.969
0.95	1742	14	0.984
0.95	970	7	0.727
0.97	472	4	0.855
0.98	197	2	1.026
0.99	60	0	0.000
1.00	7	0	0.000

#### **4.4 Discussion**

Comparing the difference between PubGene (Jenssen et al., 2001) and Poisson modelling method for co-occurrence calculations, three observations could be made. Firstly, one of the common criticisms of a simple co-occurrence method as used in this

study (co-occurrence of terms without considering the number of words between these terms) is that given a large number of articles or documents, every term will co-occur with every term at least once, leading to total possible co-occurrence (100% or 12694969 in this case). Our results showed that 7.31% of the total possible co-occurrence were actually found using about 860000 abstracts and only 3.40% using a more stringent method. PubGene (Jenssen et al., 2001) has also suggested that total possible co-occurrence was not evident with a much larger set of articles (10 million) and yet achieved 60% precision using only one instance of co-occurrence in 10 million articles (1-Mention PubGene) and 72% precision with 5-Mention PubGene. It can be expected with more instances of co-occurrence, precision may be higher. This might be due to the sparse distribution of entity names in the set of text as observed from the low mean number of co-occurrence used for Poisson distribution modeling. At the same time, PubGene (Jenssen et al., 2001) also illustrated that entity name recognition by simple pattern matching is able to yield quality results.

Using only results from PubGene (Jenssen et al., 2001), it can be concluded that total possible co-occurrence is unlikely for a corpus size of up to 10 million (more than half of current PubMed). Using the Poisson distribution, the mean number of co-occurrence can be expected to decrease with a larger corpus than used in this study as it is a product of the relative frequencies of each of the 2 entities. This suggests that as the size of corpus increases, it is likely that each co-occurrence of terms is more significant, suggesting that a statistical measure might be more useful in a very large corpus of more than 10 million as it takes into account both frequencies and corpus size.

Secondly, Poisson co-occurrence methods at both 95<sup>th</sup> and 99<sup>th</sup> percentile yield the same set of results as 1-Mention PubGene method, which is expected as the maximum mean number of co-occurrence is 0.59. This implied that every co-occurrence found are essentially statistically significant in a corpus of about 860000 abstracts; thus, providing statistical basis for “1-Mention PubGene” method. This might be due to the nature of abstracts, which were known to be concise. Proteins that have no relation to each other are generally unlikely to be mentioned in the same abstract and abstracts tends to mention only crucial findings. However, the same might not apply if full text articles are used – un-related proteins could be used solely for illustrative purposes.

Thirdly, the number of co-occurrences found using 5-Mention PubGene method is substantially lower (less than half) of that by 1-Mention PubGene method which was also shown in Jenssen et al. (2001). This suggested that 5-Mention PubGene is appreciably more stringent than using Poisson co-occurrence at 99<sup>th</sup> percentile; thus, providing statistical basis for “5-Mention PubGene” method.

Our results comparing the numbers of co-occurrence demonstrated a 50.79% decrease in co-occurrence from 1-Mention PubGene network to 5-Mention PubGene network. However, the 5-Mention PubGene network retained most of the “activation” (98.5%) and “binding” (98.0%) interactions found in 1-Mention PubGene network. This might be the consequence of 30% recall of the NLP methods (Ling et al., 2007) as it would usually require 3 or more mentions to have a reasonable chance to be identified by NLP methods. This might also be due to the observation that the 5-Mention PubGene method is more precise, in terms of accuracy, than the 1-PubGene method as shown in Jenssen et al. (2001).

The probability of a true interaction (Ling et al., 2007) existing in each of the 9661 NLP-extracted binding interactions that are also found in 1-Mention PubGene co-occurrence would be raised. The probability of a true interaction existing in each of the 9465 NLP-extracted binding interactions that are also found in 5-Mention PubGene co-occurrence would be higher. Hence, combining NLP and statistical co-occurrence techniques can improve the overall confidence of finding true interactions. However, it should be noted that statistical co-occurrence used in this work cannot raise the confidence of NLP-extracted interactions.

Nevertheless, these results also suggest that graphs of statistical co-occurrence could be annotated with information from NLP methods to indicate the nature of such interactions. In this study, 2 NLP-extracted interactions from Ling et al. (2007), “binding” and “activation”, were combined. The combined “binding” and “activation” network covered 1.96% and 3.85% of 1-Mention and 5-Mention PubGene co-occurrence graph respectively. Our results demonstrate that the combined network has a higher coverage than individual “binding” or “activation” networks. Thus, it can be

reasonable to expect that with more forms of interactions, such as degradation and phosphorylation, extracted with the same NLP techniques, the co-occurrence graph annotation would be more complete.

By overlapping the co-expression network analyzed from Master et al. (2002) data set to 1-Mention PubGene co-occurrence network, our results demonstrated that about 99% of the co-expression was not found in the co-occurrence network. This might suggest that the choice of Pearson's correlation coefficient threshold of more than 0.75 and less than -0.75 as suggested by Reverter et al. (2005) is likely to be sensitive in isolating functionally related genes from microarray data at the cost of reduced specificity.

Our results from incremental stepwise analysis showed that the percentage of overlap between co-expression and co-occurrence rose linearly from correlation coefficient from 0.75 to 0.85. This suggests that a correlation coefficient of 0.85 may be optimal for this data set as it is likely that using the correlation coefficient of 0.85 will result in less false positives than the correlation coefficient of 0.75. At the same time, increasing the correlation coefficient from 0.75 to 0.85 resulted in 77.4% less (47453 correlations from 210283) interaction correlations. Using this method to further describe protein-protein interactions and to generate new hypotheses, it can be argued that correlation coefficient of 0.85 will result in less false positives. While this deduction is likely as a more stringent criterion tends to reduce the rate of false positives, it is difficult to prove experimentally without exhaustive examination of each result. Nevertheless, the result suggests the possibility of using the inverse linearity of correlation coefficient and the number of gene co-expressions as a preliminary visual assessment to gauge an optimal correlation coefficient to use for a particular data set. However, on the extreme end, a correlation coefficient of 0.99 and 1.00 yielded 60 and 7 correlations respectively in Master et al. (2002) data set but none was found in 1-Mention PubGene co-occurrence network. This suggests that high-throughput genomic techniques such as microarrays, present a vast amount of un-mined biological information that had not been examined experimentally.

By exploring the literature for the biological significance for each of the 7 pairs of perfectly co-expressed genes using Swanson's method (Swanson, 1990), it was found

that all 7 pairs were biologically significant. Lactotransferrin (Ishii et al., 2007) and solute carrier family 3 (activators of dibasic and neutral amino acid transport), member 2 (Feral et al., 2005) were involved in cell adhesion. B-cell translocation gene 3 (Guehenneux et al., 1997) and UDP-Gal:betaGlcNAc beta 1,4- galactosyltransferase, polypeptide 1 (Mori et al., 2004) were involved in cell cycle control. Casein gamma and casein alpha are well-established components of milk. Gamma-glutamyltransferase 1 (Huseby et al., 2003) and programmed cell death 4 (Frankel et al., 2008) were known to be regulating apoptotic pathways. Rab18 (Vazquez-Martinez et al., 2007), signal recognition particle 9 (Egea et al., 2004) and FK506 binding protein 11 (Dybkaer et al., 2007) were known to be involved in the secretory pathway. G protein-coupled receptor 83 (Lu et al., 2007) and recombination activating gene 1 activating protein 1 (Igarashi et al., 2001) were known to be involved in T-cell function. Taken together, these suggest that the set of 7 correlations have not likely been described and may prove to be valuable new hypotheses in the study of mouse mammary physiology. It is also plausible that this argument can be extended to the set of 53 highly co-expressed genes ( $0.99 < \text{correlation coefficient} < 1.00$ ). This may suggest that microarray co-expression statistics may be implied as a measure of “interestingness” to rank potential interactions that had not been previously described.

Intersecting the 4 in vivo data sets into a co-expression network increases the power of the analysis as it represents correlation among gene expression that are more than 0.75 or less than -0.75 in all 4 data sets. There were 1140 examples of co-expression in this intersect and only 14 co-expressions (1.23%) were found in the one-mention PubGene co-occurrence network, but none in either the binding or activation networks extracted by natural language processing. This suggests that these 14 co-expressions are neither binding nor activating interactions. Textpresso (Muller et al., 2004) had defined a total of 36 molecular associations between 2 proteins which includes binding and activation. Future work will expand NLP mining to 34 other interactions to improve the annotation of co-occurrence networks.

Reverter et al. (2005) had previously analysed 5 microarray data sets by expression correlation and demonstrated that genes of related functions exhibit a similar expression profile across different experimental conditions. Our results suggest 1126 co-expressed

genes across 4 microarray data sets are not found in the co-occurrence network. This may be a new set of valuable information in the study of mouse mammary physiology as these pairs of genes have not been previously mentioned in the same publication and experimental examination of these potential interactions is needed to understand the biological significance of these co-expressions.

#### **4.5 Conclusions**

We conclude that the 5-mention PubGene method is more stringent than the 99<sup>th</sup> percentile of Poisson distribution method. In this study, we demonstrate the use of a liberal co-occurrence-based literature analysis (1-Mention PubGene method) to represent the state of research knowledge in functional protein-protein interactions as a sieve to isolate potentially novel hypotheses from microarray co-expression analyses for further research.

## **Chapter 5**

### **The transcriptomics of lactogenesis in the murine mammary explant model; comparison with in vivo data**

#### **5.1 Introduction**

The process of synthesizing and secreting milk is known as lactation. The onset of lactation, lactogenesis, can be divided into 2 phases: lactogenesis I, the initiation of secretory differentiation (Hartmann, 1973) and lactogenesis II, the onset of copious milk secretion (Neville and Morton, 2001). This is followed by lactation which is the continuation of copious milk production and secretion (Pang and Hartmann, 2007). Anderson et al. (2007) has termed lactogenesis II and lactation as secretory activation.

Early studies reviewed by Topper and Freeman (1980) had demonstrated that hormones have a primary role in lactogenesis I, particularly a decrease in the level of progesterone and the elevation of prolactin levels in the serum just prior to parturition. The prolactin level is elevated in the serum throughout lactation in the mouse (Hart, 1972) which is consistent with its requirement for secretory activation (Lkhider et al., 2001). Neuronal stimulation by the suckling of newborn pup triggers the release of oxytocin from the neurons located in the posterior pituitary gland resulting in milk letdown by the mammary gland (Jirikowski, 1992). The subsequent removal of milk from the mammary gland is a requirement for ongoing lactation (Davis et al., 1998; Hendry et al., 1998; Peaker and Wilde, 2006; Wall and McFadden, 2008); therefore, the neuronal stimulus is essential for secretory activation. This suggests that the lactogenesis II in the mouse is the result of a combination of stimuli by the neuronal and hormonal systems. Early studies using organ-ablated rats to show a subsequent loss of lactation (Cowie et al., 1960; Cowie and Tindal, 1961; Nicholas and Hartmann 1981b), despite the presence of suckling, confirmed an essential role of hormones in lactogenesis II. A number of mammary explant studies (Chomczynski et al., 1986; Ganguly et al., 1979; Nagaiah et al., 1981; Topper et al., 1980a; Topper et al., 1980b; Yoshimura and Oka, 1990)



demonstrated a combination of insulin (I), prolactin (P), and glucocorticoid (F) induced the expression of genes for milk proteins such as  $\beta$ -casein. The expression of milk protein genes is required for milk synthesis (Anderson et al., 2007; Juergens et al., 1965; Rosen et al., 1982) suggesting that milk protein genes such as  $\beta$ -casein can be used as a marker gene for lactogenesis I. However, it has not been shown that IFP-induced mouse mammary explant is a suitable model for the in vitro study of the more comprehensive process of mouse lactogenesis I.

A series of mammary explant experiments demonstrated that prolactin is needed for milk protein gene expression (Devinoy et al., 1978; Terry et al., 1977). Prolactin activates the JAK/STAT pathway (Groner, 2002) through a membrane-bound receptor which is positively regulated by physiological levels of prolactin (Djiane et al., 1979) and has an essential role in the transcription of the  $\beta$ -casein gene (Chomczynski et al., 1986; Kulski et al., 1983a; Kulski et al., 1983b). At the same time, heterodimerization of STAT5a/STAT5b, from the activation of JAK/STAT pathway following stimulation by prolactin, was known to activate several transcription factors necessary for expression of mammary genes (Malewski et al., 2002; Zhao et al., 2002). Using the mammary explant culture model, Chomczynski et al. (1986) demonstrated that glucocorticoid also played a role in the transcription and stability of  $\beta$ -casein mRNA. This result might have been masked in previous experiments due to retention of glucocorticoid in the mammary explants (Bolander et al., 1979; Chomczynski et al., 1986). Insulin, which was originally thought to play a role in maintaining cell viability and hormone-responsiveness of explants in vitro (Nicholas et al., 1983), has also been shown to be essential for the transcription of  $\beta$ -casein gene (Chomczynski et al., 1986) and the stability of  $\beta$ -casein mRNA in the presence of prolactin (Choi et al., 2004). In addition, a recent study by Brennan et al. (2008) demonstrated the ability of explants to remain responsive to hormones after culture in the absence of insulin supported the view that insulin does not play a role in maintaining hormone-responsiveness of explants in culture. In addition to the role of insulin in the expression milk protein genes and the stability of the transcripts, insulin has been shown to play an important role in nutrient uptake into the mammary gland (Bequette et al., 2001) and the stimulation of milk protein synthesis (Menzies et al., 2009), as well as making mammary epithelial cells more receptive to other hormones, such as growth hormone (Butler et al., 2003).

This experimental evidence suggested that the mouse mammary explant culture model is a suitable tool to study the role of hormones involved in lactogenesis I using a marker gene approach such as  $\beta$ -casein. However, the ability of the explants to secrete milk proteins (Barash et al., 1995; Yoshimura and Oka, 1990) and the proliferation of mammary epithelial cells (Keys et al., 1994) is limited. This suggests that the mammary explant model might not be suitable for the study of secretory activation, comprising of lactogenesis II and lactation.

Although the marker gene approach had demonstrated the necessity of IFP in lactation, it is limited in the number of genes studied. Microarray technology allows for the analysis of thousands of genes in a single experiment and had been used to study various stages of the mouse lactation with good success. For example, Master et al. (2002) demonstrated that different sets of genes are up-regulated at different phases of mouse mammary gland development from the onset of puberty to adult, pregnancy, and from early lactation to involution. Clarkson and Watson (2003) demonstrated down-regulation of STAT 5 and up-regulation of STAT 3 during involution. Inflammation response genes were up-regulated throughout involution (Clarkson and Watson, 2003; Stein et al., 2003) which co-incides with the activation of neutrophils into the mouse mammary gland at the initiation of involution (Clarkson and Watson, 2003) and the activation of macrophages and eosinophils are required in the later stage of involution (Stein et al., 2003). Rudolph et al. (2007) used microarrays to study the effects of diets of various fat content on lipid synthesis in the mouse mammary gland and found that the mammary gland is primarily responsible for the synthesis and secretion of milk components while the liver is responsible for responding to dietary fat content. Most importantly, it was collectively demonstrated (Master et al., 2002; Stein et al., 2003; Clarkson and Watson, 2003; Rudolph et al., 2007) that the cyclic nature of mammary gland development during pregnancy and lactation is reflected at the transcriptional level, suggesting that microarray is a suitable tool to study various aspects of mouse lactation.

In this chapter, the transcriptome of mouse mammary gland at lactogenesis and IFP-induced mammary explants from pregnant mice was compared to address the question

of whether the mammary explants mimic the *in vivo* lactogenic response. Our results demonstrated that the transcriptional profile of mouse mammary explants after 2 days of IFP-stimulation is significantly different from the transcriptional profile of the *in vivo* mammary gland at day 1 lactation suggesting that IFP-induced mammary explants do not mimic the *in vivo* lactogenic response.

## **5.2 Materials and Methods**

### **5.2.1 Mice**

Mice (CV40) were obtained from the Howard Florey Institute of The University of Melbourne, Parkville, Melbourne. Mice were maintained in the Zoology Department animal house, feed and water provided *ad libitum*. Day one of pregnancy was identified as the first day a postcoital plug was observed. Mice were euthanized on day 12 of pregnancy and their inguinal and abdominal mammary glands excised under sterile conditions.

### **5.2.2 Tissue Culture**

Mammary gland explants from mid-pregnant mice were prepared and cultured in Medium 199 as described previously (Nicholas et al., 1991), except that 10% foetal calf serum (FCS) was added and bovine serum albumin was excluded from the media. Briefly, explants were incubated at 37°C and 5% CO<sub>2</sub> in 5ml of media per well in 6-well plates, and the media changed every third day. Hormones were added at the following concentrations in the indicated combinations: bovine insulin (I; 100ng/ml, Sigma), hydrocortisone (F; 50ng/ml, Sigma), and ovine prolactin (P; 200ng/ml, National Hormone and Pituitary Program USA NIDDK-oPRL-21). All explants were initially cultured in the absence of exogenous hormones (NH) for eight days to allow effects of endogenous hormones and inflammatory response of the tissue to subside. At day eight mammary explants were harvested for controls, and the remaining explants cultured for either two or four days in media containing FP, IP, IF and IFP. Mammary glands from a total of 12 mice were used in three separate culture experiments with mammary glands from four mice pooled for each culture experiment. In each experiment, three wells of explants were cultured for each hormone treatment and NH control. Explants were

collected and stored at -80°C until RNA was extracted for microarray analysis.

### **5.2.3 Microarray – Hybridisation and Analysis**

Total RNA was extracted using an RNeasy Lipid Tissue Mini-kit (Qiagen) following the manufacturer's instructions. The cDNA was synthesised from total RNA (20µg) using Superscript II (Invitrogen Life Technologies). Synthesis of Biotin-labelled cRNA was performed using BioArray High Yield RNA Transcript labeling Kit (Enzo Diagnostics). The cDNA probes were hybridised to Affymetrix MOE430 GeneChips overnight according to manufacturer's instructions. Five arrays were performed in total, each microarray chip representing one treatment using pooled RNA from each of the three mammary explant culture experiments.

Initial analysis was performed using the Affymetrix GeneChip® Operating Software (GCOS) to assess array quality. Signal intensities of each gene were obtained using the robust multi-array average (RMA) function of the Affy package in bioconductor (<http://www.bioconductor.org>). RMA analysis involves quantile normalization of oligonucleotide signals followed by estimation of the average perfect match signal intensity for each probe-set, and has been shown to reduce variability and bias when compared to the MAS 5.0 software (Irizarry et al., 2003).

### **5.2.4 Microarray Datasets**

Three normalized microarray datasets consisting of 2 in vivo and 1 in vitro datasets were examined, including the in vitro dataset described in Sections 5.2.1-5.2.3 above. The 2 in vivo datasets were from Rudolph et al. (2007) using Affymetrix U74Av2 chip and FVB mice, and Stein et al. (2003) using Affymetrix U74Av2 chip and Balb/C mice.

### **5.2.5 Assessing the Mammary Explant Model with Marker Genes**

The hormone-dependent response of the mouse mammary explant model was evaluated by examining the expression of 5 classes of marker genes. Caseins are components of milk protein (Juergens et al., 1965) which can be used as markers for milk protein synthesis in lactogenesis I and II. Acyl-coA synthetase activates fatty acids for lipid

synthesis (Jia et al., 2007) and can be used as marker for milk lipid synthesis in lactogenesis I and II. Insulin (Taha and Klip, 1999) and prolactin (Hennighausen et al., 1997) signaling pathway components are required for signal transduction and can be used as markers for insulin and prolactin responsiveness in lactogenesis I and II. Cell surface receptors consisting of growth hormone, nutrient (folate) and signal (activin and TGF-beta) receptors are as markers of signal responsiveness of the mammary explants in lactogenesis I. In addition, the growth hormone receptor had been shown to be up-regulated by insulin (Butler et al., 2003) suggesting the use of growth hormone receptor as a marker of insulin response. Folate is a component in milk and folate receptor may be used as a marker for milk secretion in lactogenesis II (Holm et al., 2000). Fatty acid binding proteins are required for the transport of fatty acid synthesized and/or up-taken by the mammary epithelial cells (Rudolph et al., 2007) during milk secretion in lactogenesis II. A gene was determined to be up-regulated between time A and time B if the spot intensity in time B is 2 times or more compared to time A, and down-regulated if the spot intensity in time A is 2 times or more compared to time B.

### **5.2.6 Comparison by Statistical Analysis and Hierarchical Clustering of Transcriptomics Data**

Hierarchical clustering was carried out using the data published by Stein et. al. (2003) and the mammary explant (in vitro) experiments using Spearman's coefficients of correlation. The two microarray data sets had been normalized together using the Lowess method (Cleveland, 1981). Statistical analysis was carried out using the two-sided Wilcoxon exact signed-rank test with continuity correction on the ratios of microarray readings at 6 different time points: 17.5 days pregnancy (P17.5) and day 1 lactation (Lact1), P17.5 and day 3 lactation (Lact3), 12.5 days pregnancy (P12.5) and Lact1, P12.5 and Lact3, S8 and SIFP10, and S8 and SIFP12. The 4 former comparisons (P17.5/Lact1, P17.5/Lact3, P12.5/Lact1 and P12.5/Lact3) were carried out using Stein et. al. (2003) data set and the latter 2 (S8/SIFP10 and S8/SIFP12) were the mammary explant data set.

### **5.2.7 Comparison by Gene Ontological Clustering**

A list of genes that were up-regulated (IFP-up) following treatment of the explants with

IFP was generated by identifying the intensity values in the dataset that showed a 100% increase in either SIFP10 or SIFP12 when compared with S10 and S12 respectively. A list of genes that were up-regulated at parturition (LACT-up) was generated using the following criteria: either 100% increase in intensity values of the average or individual mouse between day 17.5 pregnancy and day 1 lactation in the Stein et al. (2003) dataset or 100% increase in intensity values of the average or individual mouse between day 17 pregnancy and day 1 lactation in the Rudolph et. al. (2007) dataset. Each of these lists (IFP-up and LACT-up) were clustered to Gene Ontology (Biological Process) and analyzed statistically for over-representation by a hypergeometric test with Benjamini and Hochberg false discovery rate (FDR) correction at 95% confidence using BiNGO (Maere et al., 2005) version 2.0 using Gene Ontology version January 12th 2007 and visualized in Cytoscape (Shannon et al., 2003) version 2.4.1.

## **5.3 Results**

### **5.3.1 Assessing the Explant Model with Marker Genes**

Five classes of marker genes were used to assess the similarity of IFP-induced mammary explants with mammary glands at parturition and lactogenesis (Table 5a). For the explant cultures, microarray data from S8, SIFP10 and SIFP12 were analyzed and comparisons were made between S8 and SIFP10, and SIFP10 and SIFP12. Microarray data from 17.5 days pregnancy (P17.5), day 1 lactation (Lact1) and day 3 lactation (Lact3) were examined and comparisons were made between P17.5 and Lact1, and Lact1 and Lact3. Results from the marker gene approach in vivo were as follows: casein, acyl-coA synthetase, folate receptor I, TGF-beta receptor III, JAK3 and STAT 5b were consistently up-regulated from day 17.5 pregnancy to day 3 lactation; growth hormone receptor and PI3K regulatory subunit 2 were consistently down-regulated from day 17.5 pregnancy to day 3 lactation; activin receptor I was up-regulated from day 17.5 pregnancy to day 1 lactation and remained constant from day 1 lactation to day 3 lactation; FABP 3 and FABP4 were down-regulated from day 17.5 pregnancy to day 1 lactation; FABP3 remained constant from day 1 lactation to day 3 lactation but FABP4 was up-regulated from day 1 lactation to day 3 lactation.

The expression patterns of casein, acyl-coA synthetase and folate receptor I in explants

were similar to in vivo results. FABP 3 and 4 were consistently up-regulated throughout the 4 days of IFP-stimulation. Activin receptor I was consistently up-regulated throughout the 4 days of IFP-stimulation. Growth hormone receptor, TGF-beta receptor III, JAK 1, STAT 5a, PI3K regulatory subunit 1 and MAPP4 were up-regulated in explants during the first 2 days of IFP-stimulation but down-regulated in the next 2 days of IFP-stimulation.

Table 5a. Summary of Gene Expression Trends between IFP-induced Mouse Mammary Explants and in vivo mammary Glands using 5 Classes of Marker Genes. “+” indicates that the gene is up-regulated across the two time points indicated. “-” indicates that the gene is down-regulated across the two time points and “NC” indicates that the gene is unchanged (not up-regulated nor down-regulated) across the two time points.

In Vitro			In Vivo		
Groups and Gene Names	Treatments		Groups and Gene names	Treatments	
	S8/ SIFP10	SIFP10/ SIFP12		P17.5/ Lact1	Lact1/ Lact3
Class 1: Casein Genes					
Alpha casein	+	+	Beta casein	+	+
Gamma casein	+	+	Gamma casein	+	+
Class 2: Acyl-CoA Synthethase					
Acyl-CoA synthethase long chain	+	+	Acyl-CoA synthethase long chain	+	+
Acyl-CoA synthethase short chain	+	+			
Class 3: Fatty Acid Binding Proteins (FABP)					
FABP 3	+	+	FABP 3	-	NC
FABP 4	+	+	FABP 4	-	+
Class 4: Cell Surface Receptor					
Growth hormone receptor	+	-	Growth hormone receptor	-	-
Folate receptor I	+	+	Folate receptor I	+	+
Activin receptor I	-	-	Activin receptor I	+	NC
TGF-beta receptor III	+	-	TGF-beta receptor III	+	+
Class 5: Prolactin and Insulin Signaling Pathway Components					
JAK 1	+	-	JAK 3	+	+
STAT 5a	+	-	STAT 5b	+	+
PI3K regulatory subunit 1	+	-	PI3K regulatory subunit 2	-	-

In Vitro		In Vivo	
Groups and Gene Names	Treatments	Groups and Gene names	Treatments
MAKK4	+ -		

### 5.3.2 Comparison by Statistical Analysis and Hierarchical Clustering of Transcriptomics Data

Using pair-wise testing of the 6 ratios (P17.5 versus Lact1, P17.5 versus Lact3, P12.5 versus Lact1, P12.5 versus Lact3, S8 versus SIFP10 and S8 versus SIFP12) by exact Wilconox signed rank test with continuity correction, the differences between the ratios within in vivo and in vitro transcriptional profiles were not statistically significant. The 4 former comparisons (P17.5/Lact1, P17.5/Lact3, P12.5/Lact1 and P12.5/Lact3) were carried out using the data set published by Stein et. al. (2003). The latter 2 comparisons (S8/SIFP10 and S8/SIFP12) were carried out using the microarray results from this study. However, all 8 comparisons between in vivo and in vitro gene expression ratios were highly significant. The p-values were summarized in Table 5b.

Table 5b: P-value Table Comparing Pre- and Post- Lactogenic Trigger in vitro and in vivo using Exact Wilconox Signed Rank Test with Continuity Correction.

	P17.5 v Lact1	P17.5 v Lact3	P12.5 v Lact1	P12.5 v Lact3	S8 v SIFP10	S8 v SIFP12
P17.5 v Lact1	1					
P17.5 v Lact3	0.8861	1				
P12.5 v Lact1	0.8653	0.4460	1			
P12.5 v Lact3	0.3096	0.9450	0.9630	1		
S8 v SIFP10	1.665e-08	1.597e-09	2.629e-06	9.817e-08	1	
S8 v SIFP12	1.941e-10	2.540e-11	1.849e-07	6.409e-09	0.2857	1

Hierarchical clustering using Spearman's coefficient of correlation was performed on a merged table of mouse mammary explant data set and Stein et. al. (2007) data set. Results showed that the mouse mammary explant clustered separately to Stein et. al. (2007)'s data with no overlapping samples between the in vivo and in vitro data sets. The dendrogram was given in Figure 5a.





Table 5c. Lists of Gene Ontologies that were Common and Singly in IFP-up and LACT-up.

p-value	GOID	GO Description	p-value	GOID	GO Description
Gene Ontologies that are common in both IFP-up and LACT-up					
6.56E-023	8151	cellular physiological process	2.08E-005	6950	response to stress
2.23E-013	7275	development	3.33E-005	2245	physiological response to wounding
2.23E-013	48856	anatomical structure development	3.35E-005	9611	response to wounding
1.01E-008	51234	establishment of localization	9.52E-003	9792	embryonic development (sensu Metazoa)
1.64E-008	51179	localization	1.74E-002	6954	inflammatory response
9.56E-008	8152	metabolism	2.45E-002	16192	vesicle-mediated transport
2.20E-007	9653	morphogenesis	2.76E-002	43549	regulation of kinase activity
7.47E-007	48513	organ development	3.51E-002	51338	regulation of transferase activity
5.48E-006	6810	transport	4.09E-002	43009	embryonic development (sensu Vertebrata)
7.41E-006	7582	physiological process			
9.46E-006	9605	response to external stimulus			
Gene Ontologies that are only in IFP-up					
4.64E-002	14032	neural crest cell development	2.77E-002	45859	regulation of protein kinase activity
4.64E-002	14033	neural crest cell differentiation	2.77E-002	9889	regulation of biosynthesis
4.49E-002	2526	acute inflammatory response	2.77E-002	7155	cell adhesion
4.49E-002	6575	amino acid derivative metabolism	2.05E-002	42136	neurotransmitter biosynthesis
4.14E-002	48514	blood vessel morphogenesis	1.92E-002	6817	phosphate transport
4.14E-002	16485	protein processing	1.34E-002	16193	endocytosis
3.46E-002	45927	positive regulation of growth	1.34E-002	1568	blood vessel development
3.40E-002	16540	protein autoproccessing	1.34E-002	1944	vasculature development
3.35E-002	46777	protein amino acid autophosphorylation	1.34E-002	1701	embryonic development (sensu Mammalia)
3.05E-002	45429	positive regulation of nitric oxide			
Gene Ontologies that are only in LACT-up					
1.48E-008	9987	cellular process	1.27E-002	79	regulation of cyclin-dependent protein kinase
6.03E-008	44237	cellular metabolism	1.27E-002	48729	tissue morphogenesis
8.01E-008	6260	DNA replication	1.29E-002	48771	tissue remodeling
1.58E-007	30154	cell differentiation	1.29E-002	15031	protein transport
8.75E-007	43119	positive regulation of physiological process	1.33E-002	30236	anti-inflammatory response
1.39E-006	9888	tissue development	1.33E-002	6269	DNA replication, synthesis of RNA primer
1.47E-006	51242	positive regulation of cellular physiological	1.34E-002	6091	generation of precursor metabolites and energy
1.71E-006	44238	primary metabolism	1.46E-002	51186	cofactor metabolism
2.93E-006	48522	positive regulation of cellular process	1.63E-002	6631	fatty acid metabolism
2.93E-006	44255	cellular lipid metabolism	1.64E-002	46907	intracellular transport
3.28E-006	6629	lipid metabolism	1.64E-002	51641	cellular localization
3.68E-006	48731	system development	1.74E-002	165	MAPKKK cascade
3.70E-006	8610	lipid biosynthesis	1.76E-002	48609	reproductive organismal physiological process
4.98E-006	8283	cell proliferation	1.76E-002	7346	regulation of progression through mitotic cell
5.22E-006	48518	positive regulation of biological process	1.80E-002	50819	negative regulation of
7.41E-006	9058	biosynthesis			

7.83E-006	9887	organ morphogenesis			coagulation
8.97E-006	19752	carboxylic acid metabolism	1.80E-002	50794	regulation of cellular process
9.27E-006	6082	organic acid metabolism	1.89E-002	50876	reproductive physiological process
1.84E-005	16126	sterol biosynthesis			
2.08E-005	8202	steroid metabolism	1.95E-002	50673	epithelial cell proliferation
2.79E-005	7596	blood coagulation	1.95E-002	9064	glutamine family amino acid metabolism
3.09E-005	7399	nervous system development			
3.16E-005	6807	nitrogen compound metabolism	1.95E-002	48523	negative regulation of cellular process
3.17E-005	50817	coagulation	1.98E-002	48193	Golgi vesicle transport
3.60E-005	16125	sterol metabolism	2.00E-002	44262	cellular carbohydrate metabolism
4.31E-005	6694	steroid biosynthesis			
4.59E-005	6066	alcohol metabolism	2.01E-002	7610	behavior
5.92E-005	7599	hemostasis	2.09E-002	43118	negative regulation of physiological process
5.92E-005	42127	regulation of cell proliferation			
5.92E-005	9308	amine metabolism	2.14E-002	8219	cell death
7.22E-005	6519	amino acid and derivative metabolism	2.18E-002	16265	death
			2.18E-002	12501	programmed cell death
9.28E-005	42060	wound healing	2.32E-002	1707	mesoderm formation
1.09E-004	6695	cholesterol biosynthesis	2.37E-002	6915	apoptosis
1.96E-004	50878	regulation of body fluids	2.37E-002	30500	regulation of bone mineralization
2.22E-004	8203	cholesterol metabolism			
3.19E-004	30855	epithelial cell differentiation	2.38E-002	45941	positive regulation of transcription
3.66E-004	50791	regulation of physiological process			
6.00E-004	9063	amino acid catabolism	2.43E-002	7398	ectoderm development
8.75E-004	6520	amino acid metabolism	2.43E-002	6004	fucose metabolism
9.98E-004	8284	positive regulation of cell proliferation	2.43E-002	50818	regulation of coagulation
			2.43E-002	8104	protein localization
9.98E-004	46903	secretion	2.43E-002	9074	aromatic amino acid family catabolism
1.02E-003	7267	cell-cell signaling	2.43E-002	46890	regulation of lipid biosynthesis
1.06E-003	6558	L-phenylalanine metabolism	2.43E-002	19218	regulation of steroid metabolism
1.23E-003	31214	biomineral formation			
1.50E-003	6486	protein amino acid glycosylation	2.43E-002	45893	positive regulation of transcription, DNA-dependent
1.68E-003	30501	positive regulation of bone mineralization	2.43E-002	278	mitotic cell cycle
1.68E-003	6572	tyrosine catabolism	2.65E-002	48332	mesoderm morphogenesis
1.68E-003	6273	lagging strand elongation	2.65E-002	8544	epidermis development
2.17E-003	9790	embryonic development	2.87E-002	50678	regulation of epithelial cell proliferation
2.20E-003	6270	DNA replication initiation	2.87E-002	5975	carbohydrate metabolism
2.20E-003	7595	lactation	2.91E-002	9893	positive regulation of metabolism
2.55E-003	43413	biopolymer glycosylation			
3.34E-003	9310	amine catabolism	2.91E-002	45935	positive regulation of nucleobase, nucleoside
3.37E-003	1503	ossification			
3.37E-003	40011	locomotion	3.00E-002	35295	tube development
3.37E-003	6725	aromatic compound metabolism	3.01E-002	43010	eye development (sensu Vertebrata)
3.63E-003	44270	nitrogen compound catabolism	3.15E-002	1704	formation of primary germ layer
3.63E-003	6263	DNA-dependent DNA replication	3.16E-002	7049	cell cycle
3.64E-003	44249	cellular biosynthesis	3.20E-002	48468	cell development
3.65E-003	6093	carbohydrate biosynthesis	3.26E-002	42559	pteridine and derivative biosynthesis
3.65E-003	9101	glycoprotein biosynthesis	3.26E-002	7498	mesoderm development
3.93E-003	6271	DNA strand elongation	3.40E-002	45944	positive regulation of transcription from RNA
4.20E-003	51674	localization of cell			
4.20E-003	6928	cell motility	3.51E-002	1755	neural crest cell migration
5.02E-003	6886	intracellular protein transport	3.54E-002	16477	cell migration

5.02E-003	50789	regulation of biological process	3.57E-002	7507	heart development
5.06E-003	1501	skeletal development	3.62E-002	7420	brain development
6.46E-003	7417	central nervous system development	3.65E-002	9072	aromatic amino acid family metabolism
6.71E-003	2009	morphogenesis of an epithelium	3.66E-002	6259	DNA metabolism
6.86E-003	6888	ER to Golgi vesicle-mediated transport	3.77E-002	19318	hexose metabolism
7.26E-003	48519	negative regulation of biological process	3.78E-002	279	M phase
7.41E-003	6527	arginine catabolism	3.78E-002	51253	negative regulation of RNA metabolism
7.41E-003	6559	L-phenylalanine catabolism	3.78E-002	51046	regulation of secretion
7.41E-003	51244	regulation of cellular physiological process	3.78E-002	50686	negative regulation of mRNA processing
7.41E-003	46852	positive regulation of bone remodeling	3.78E-002	15810	aspartate transport
7.41E-003	45778	positive regulation of ossification	3.78E-002	30195	negative regulation of blood coagulation
7.68E-003	19721	pteridine and derivative metabolism	3.78E-002	7497	posterior midgut development
7.68E-003	43170	macromolecule metabolism	3.78E-002	30193	regulation of blood coagulation
7.77E-003	46849	bone remodeling	3.78E-002	7435	salivary gland morphogenesis
7.95E-003	45184	establishment of protein localization	3.82E-002	9056	catabolism
7.95E-003	9100	glycoprotein metabolism	3.94E-002	51325	interphase
8.18E-003	30282	bone mineralization	3.94E-002	51329	interphase of mitotic cell cycle
9.52E-003	45045	secretory pathway	4.03E-002	9065	glutamine family amino acid catabolism
1.09E-002	6570	tyrosine metabolism	4.03E-002	31424	keratinization
1.09E-002	6487	protein amino acid N-linked glycosylation	4.03E-002	6749	glutathione metabolism
1.09E-002	43269	regulation of ion transport	4.03E-002	19438	aromatic compound biosynthesis
1.11E-002	51649	establishment of cellular localization	4.03E-002	6525	arginine metabolism
1.11E-002	6732	coenzyme metabolism	4.23E-002	7389	pattern specification
1.14E-002	2376	immune system process	4.30E-002	50790	regulation of catalytic activity
1.21E-002	9312	oligosaccharide biosynthesis	4.53E-002	44248	cellular catabolism
			4.55E-002	6357	regulation of transcription from RNA polymerase
			4.56E-002	45454	cell redox homeostasis
			4.59E-002	5996	monosaccharide metabolism

## 5.4 Discussion

A large number of studies using mid-pregnant mouse mammary explant culture (Choi et al., 2004; Chomczynski et al., 1986; Topper and Freeman, 1980; Topper et al., 1984; Yoshimura and Oka, 1990) had shown that insulin (I), prolactin (P), and glucocorticoid (F) are required for expression of the  $\beta$ -casein gene and the accumulation of its transcripts. However, these studies did not examine whether the IFP-induced mouse mammary explant is a suitable model for the in vitro study of mouse lactogenesis I and II. Recent studies using microarray technology to study the mouse lactation cycle in vivo had demonstrated distinct sets of up-regulated genes at late pregnancy, early lactation, mid-lactation, early involution and late involution (Master et al., 2002). This

suggests that the regulation of lactation can be observed at the transcription level (Clarkson and Watson, 2003; Rudolph et al., 2007; Stein et al., 2004). These data suggest that microarray technology can be used to compare the transcriptional profile changes resulting from IFP induction of mouse mammary explants to datasets identifying changes in gene expression patterns in the mammary gland between late pregnancy and early lactation to address the question of whether IFP-induced mammary explants mimic the in vivo lactogenic responses.

We have used 5 groups of marker genes to examine lactogenesis in mammary explants and mammary tissues in late pregnancy and early lactation. Our results showed that casein and acyl-coA synthetase genes expression had a similar trend in both in vivo and in vitro models suggesting that the ability of the in vivo gland to synthesize casein and lipid is likely to be replicated in the explant model. This is consistent with previous studies demonstrating casein synthesis (Topper and Freeman, 1980) and fatty acid synthesis (Bussman et al., 1984; Forsyth and Turvey, 1984) in IFP-induced mammary explants.

FABPs had been shown to be down-regulated between late pregnancy (P17.5) and the first day of lactation in the in vivo mammary gland. This is consistent with a previous study comparing the mammary and liver transcription profiles showing up-regulation of FABPs in the liver but down-regulated in the lactating mammary gland, suggesting that the liver plays an important role in transporting fatty acids from the gastrointestinal tract to the mammary gland where fatty acids are used in milk lipid synthesis (Rudolph et al., 2007). However, our results showed that FABPs were up-regulated in explants. FABPs were also shown to be up-regulated following either nerve injury (De Leon et al., 1996) or intestinal injury (Derikx et al., 2008). This might be due to the need to utilize fatty acids for tissue repair as a result of cutting the mammary gland for explant culture which is supported by predominant biosynthetic and development related Gene Ontologies observed in IFP-up but not in LACT-up. This is consistent with a number of studies suggesting that fatty acids are needed for tissue repair (Jais et al., 1994; Kusakari et al., 2006; Pratt et al., 2001; Zakaria et al., 2007) as fatty acids is a major component of cell membrane.

Prolactin signaling proteins were up-regulated in the explants following IFP stimulus which is consistent with IFP-induced milk protein gene expression in mammary explants (Choi et al., 2004) but explants failed to sustain the up-regulation for more than 2 days. In contrast, data from the analysis of *in vivo* glands showed consistent up-regulation from late pregnancy to day 3 of lactation. This suggests that the mammary explant is unlikely to maintain responsiveness to prolactin for an extended period of time. This is consistent with a previous study on insulin by Oka et al. (1974) demonstrating reduced insulin responsiveness after more than 3 days in culture supplemented with insulin. It may be possible that the mammary explants lacked additional stimuli to maintain responsiveness to prolactin as it had been shown that mammary epithelial cells require mammary fibroblasts to maintain estrogen responsiveness in culture (Haslam and Levely, 1985). PI3k, an Insulin signaling molecule, was down-regulated *in vivo*. Lactation had been shown to be mainly a hypo-insulinemic condition compared to pregnancy (Burnol et al., 1986). A study using diabetic rats had shown that short-term insulin deficiency does not prevent the secretion of milk components (Kyriakou and Kuhn, 1973). This suggests that although insulin is required in the synthesis of milk proteins (Choi et al., 2004) and nutrient uptake into the lactating gland (Bequette et al., 2001), the requirement for insulin is reduced in lactation compared to pregnancy and this is consistent with a down-regulation of insulin signaling pathway molecules from late pregnancy (P17.5) to day 3 lactation. Another explanation for the reduced need for insulin in lactation compared to pregnancy may be the activation of insulin signaling pathway by prolactin (Abell et al., 2005). However, in the current study, insulin signaling proteins were up-regulated in explants during the first 2 days of IFP-stimulus. This may be explained by the presence of tissue injury as insulin signaling had been shown to be up-regulated in neural injury (Xu et al., 2004b). The pattern of expression of prolactin and insulin signaling proteins suggests that the IFP-induced mouse mammary explants did not maintain responsiveness to IFP stimulus for more than 2 days. This might suggest the requirement of additional stimulus to maintain IFP responsiveness in culture as previously shown in estrogen responsiveness (Haslam and Levely, 1985). Hence, the IFP-induced mouse mammary explant model might not be suitable to study the hormonal effect on the explants for a period of not more than 2 days, together with the physiological and biochemical processes that requires sustained hormonal stimuli such as secretory activation (Anderson et al., 2007).

Expression of the folate receptor I gene showed consistent up-regulation in both in vivo and in vitro models suggesting that the ability of folate uptake is likely to be replicated in the explant model. The gene expression profiles of growth hormone receptor and TGF-beta receptor III in vivo were consistent with previous studies on late pregnant mice showing an up-regulation of TGF-beta receptor III (Robinson et al., 1991) and down-regulation of growth hormone receptor (Ilkbahar et al., 1999). The gene expression profile for growth hormone receptor was consistent with the gene expression profiles of insulin signaling molecules (PI3K subunits and MAPKK) as insulin had been suggested to play a role in increasing the responsiveness of mammary epithelial cells to growth hormone (Butler et al., 2003). Our results showed that activin receptor I was up-regulated in vivo from late pregnancy is consistent with a previous study demonstrating up-regulation of activin signaling in lactation (Jeruss et al., 2003). In addition, our results showed that activin receptor I was consistently down-regulated in explants. Activin receptor has previously been shown to be activated by epidermal growth factor (EGF) in mouse epithelial cells (Bianco et al., 2002) suggesting that activin receptor can be a marker for EGF signaling. Our results suggested that EGF signaling had not been replicated in IFP-induced mouse mammary explant culture.

The statistical comparison of the gene expression profiles of in vivo glands between day 12.5 pregnancy and day 17.5 pregnancy to day 1 lactation gave a p-value of 0.8653, indicating no statistical difference between the transcriptional profiles of the mammary gland from mid-pregnancy (day 12.5 pregnancy) to late pregnancy (day 17.5 pregnancy) in the mouse. This result is consistent with a previous study by Lemay et al. (2007) which demonstrated 985 genes are up-regulated throughout pregnancy but only 122 genes are up-regulated from late pregnancy to mature lactation, suggesting a majority of the transcriptional commitments to lactation took place before parturition. In addition, Vonderhaar and Smith (1982) demonstrated that casein can be expressed in IFP-induced mouse mammary explants culture of virgin mice. This suggests that IFP-induction of mammary tissues comprised of either ductal cells only or a combination of ductal and alveolar epithelial cells is likely to result in casein gene expression. This also suggests that mammary tissues from day 12.5 pregnancy to day 17.5 pregnancy can be used in explant experiments with the expectation that the explants will exhibit the same

hormone-induced responses.

The transcriptional profiles from day 17.5 pregnant to day 1 lactation in vivo is significantly different to transcriptional profiles of the explants before and 2 days after IFP stimulation ( $p = 1.665e-8$ ). This result is consistent with the results from hierarchical clustering which showed that data sets from in vivo samples and in vitro samples clustered separately. This result also supports our results from the marker gene analysis, suggesting that gene expression changes induced by IFP explants were different to the changes in gene expressions between day 17.5 pregnant to day 1 of lactation.

Our results demonstrated a consistent trend that the transcriptional profile of explants after 4 days in culture is less similar to in vivo than after 2 days in culture. Results from the marker gene approach suggests that the explants might not be able to sustain responsiveness to IFP stimuli (Quirk et al., 1986; Quirk and Funder, 1988) for more than 2 days due to a down-regulation of prolactin and insulin signalling molecules. Hence, our results from transcriptional profile comparison suggests that the mammary explant culture model may not be suitable for physiological or biochemical studies that requires hormone stimulation of more than 2 days in culture in hormones, such as studying lactogenesis II.

Hierarchical clustering demonstrated that gene expression datasets from explants did not cluster with the in vivo samples (Figure 5a) which was expected as our statistical analysis of the transcriptional profiles had showed that these transcriptional profiles are significantly different. In addition, data from Figure 5a also demonstrated that the explants cultured without hormone did not cluster together with in vivo samples, suggesting that the transcriptional profile after the initial 8 days of culture in the no hormone environment is different from that of mid-pregnant mammary glands. Initial culture in a no hormone environment allowed for diffusion of hormones that may be retained in the mammary fat pad, especially glucocorticoid (Bolander et al., 1979; Chomczynski et al., 1986). However, the diffusion of glucocorticoid from the mammary fat pad is a concentration dependent process. In addition, casein and alpha-lactalbumin had been shown to be preferentially induced at different concentrations of



glucocorticoid (Ono and Oka, 1980) suggesting that decreasing hormone levels will effect the transcriptional and biochemical states of the explants.

Analysis of the IFP-up and LACT-up data sets suggests that the response in mammary explants is lacking a range of metabolic activities, such as lipid metabolism, steroid metabolism, cholesterol metabolism, amino acid metabolism and secretion. These processes of lipid metabolism and secretion had been suggested by Rudolph et. al. (2007) to be crucial to lactation, suggesting an incompetency of the explants to secrete. However, this response is not observed in the IFP-up genes. This is supported by study reported by Collier et al. (1977) demonstrating that IFP-induced explants have limited secretory activity. Secondly, mammary tissue from both in vivo and in explants express gene markers associated with metabolic stress and injury. Both data sets showed genes responding to stress and wounding, organ development and inflammatory responses were common in both. This is consistent with Hadsell et al. (2007) showing high level of oxidative damage to mitochondrial proteins confirming that the normal physiology of lactation is a metabolically demanding activity. Thirdly, the Gene Ontologies related to the IFP-up data set suggest an acute stress response and tissue survival activities with ontological terms such as cell adhesion, blood vessel development, positive regulation of growth. This might due to tissue injury during the cutting of explants or the culture conditions which resulted in tissue stress and repairs.

Although this study had demonstrated that the transcriptome of IFP-induced explants is different to lactating glands, it is known that IFP alone is not replicative of the entire range of circulating hormones at lactogenesis (Topper and Freeman, 1980). Other hormones reviewed by Topper and Freeman (1980) such as estrogen and thyroid hormones had been shown to be required for the synthesis of lactotransferrin (Teng et al., 1989) and  $\alpha$ -lactalbumin (Vonderhaar, 1977; Ziska et al., 1988) respectively. Other studies had also linked estrogen to lactose synthesis (Bolander and Topper, 1980) and vitamin D (Jacobson et al., 1989), IGF-1 (Kleinberg, 1997) and EGF (Schroeder and Lee, 1997) to the regulation of mammary tissue differentiation. Hence, the explant model may be a useful experimental tool for further work to more comprehensively examine the requirement of the range of circulating hormones at lactogenesis I by stepwise addition of hormones to the culture media. Therefore, using the current IFP-

induced explant model as a baseline for further additive studies will improve the understanding of the roles of other hormones at lactogenesis which will in turn provide a platform for dissecting the neuronal effects, such as suckling, on lactogenesis II.

## Chapter 6

### General Discussion

#### **6.1 Introduction**

The initiation of lactation, lactogenesis, can be divided into two phases: lactogenesis I, the initiation of secretory differentiation (Hartmann, 1973) and lactogenesis II, the onset of copious milk secretion (Neville and Morton, 2001). Early mouse studies demonstrated the importance of hormonal control over neuronal (suckling) control of lactogenesis (Nicholas and Hartmann, 1981b). Mammary explant culture had been used to elucidate the requirement for IFP on milk protein gene expression and accumulation of its transcripts (Choi et al., 2004) but it has not been shown that IFP-induced mouse mammary explant is a suitable model for the in vitro study of mouse lactogenesis. The primary advantage of explant cultures over animals is the ease of chemical and hormonal manipulation without the need of invasive procedures as shown in Nicholas and Hartmann (1981b). However, explants cannot be used to study the suckling effect or homeostatic regulation provided by other organs (Rudolph et al., 2007).

Microarray technology allowed for the analysis of thousands of gene expressions following a treatment and had been used to study different stages of mouse lactation cycle (Master et al., 2002; Clarkson and Watson, 2003; Stein et al., 2004). The comprehensive assessment of thousands of genes is a challenging task as amount of information in published form is increasing exponentially, making it difficult for researchers to keep abreast with the relevant literature as most of the research results are not found in databases (Hunter and Cohen, 2006). Computational text analysis has been used as a tool to analyze biological text to mine for specific assertions such as protein phosphorylation (Hu et al., 2005) or entity interactions (Abulaish and Dey, 2007). This research demonstrates an application of computational text analysis and microarray analysis to address the biological question of whether the IFP-induced mouse mammary explant is a suitable model for the in vitro study of mouse lactogenesis and to derive

potentially novel hypotheses for future biological research.

## **6.2 Mouse Mammary Explant Model is a Suitable Model to Study Secretory Differentiation but not Secretory Activation**

The biological question posed in this research is whether the IFP-induced mouse mammary explant is a suitable model for the in vitro study of mouse lactogenesis. Chapter 5 suggests that IFP-induced mammary explants is a suitable model to study the hormonal requirements for secretory differentiation (lactogenesis I) but may not be suitable for any study requiring more than 2 days of hormonal stimuli, such as secretory activation (lactogenesis II and lactation), as hormone responsiveness of the explants decline after 2 days post hormone stimulation.

However, the data from studies of IFP-induced mammary explants can provide a baseline to study the effects and roles of other hormones on secretory differentiation using an additive approach to bring the hormone-induced transcriptome closer to that of day 1 lactation. This hypothesis needs further verification; therefore, future work can include studies on the requirement for thyroxine (Slebozinski et al., 1999), estrogen (Ziska et al., 1988), and galanin (Naylor et al., 2005), singly and in combination with the IFP to compare the resulting transcriptomes as a proof-of-concept to this hypothesis.

Gene Ontology analysis in Chapter 5 is restricted to up-regulated genes. The analysis of down-regulated genes has not received as much attention as up-regulated genes. Lactogenesis turned “*a quiescent organ into an incredibly efficient machine for the massive synthesis and secretion of a complex mixture*” (Rudolph et al., 2007). It is plausible to conceive that pathways not essential to lactation may be down-regulated or inactivated. Hence, an examination of the down-regulated genes during secretory activation might complement our understanding of lactogenesis.

This study had demonstrated a comprehensive method of transcriptomic comparison to address the suitability of using in vitro models to the study of lactogenesis. This method may be extended to assess the suitability of mammosphere cultures to the study of lactogenesis (Hurley et al., 1994). Mammary epithelial cells cultured on collagen gels were able to form alveolus-like structures that were shown to be capable of secreting

casein into the luminal space (Hurley et al., 1994). This suggests that mouse mammosphere culture may be used to study the process of secretory activity which is lacking in mouse explant culture (Barash et al., 1995). In addition, mouse mammospheres had been transplanted into mice with their mammary glands partially removed by surgery and shown to be able to re-populate the entire mammary gland (Liao et al., 2007). This suggested that mouse mammosphere culture may be a suitable culture model for the study of mouse lactogenesis I and II.

It may be likely to use the same method presented in this study for other in vivo/in vitro comparisons such as the use of cell lines. Cell lines have been used as an immortalized and stable substitute for primary cultures in many biological studies. However, do cell lines mimic in vivo tissues or primary cultures?

### **6.3 *Computational Linguistics can Annotate Statistical Linguistics***

Computational text processing methods can be generally classified into 2 categories (Cohen and Hunter, 2008): natural language processing or computational linguistics and statistical linguistics. Natural language processing is based on the English grammar (Rindflesch et al., 2000) whereas statistical linguistics is based on the proximity of words or the relative probability of words (Cohen and Hunter, 2008). The performance of different computation text processing systems (Tsai et al., 2006) is measured as precision or specificity and recall or sensitivity. Generally, statistical linguistics achieves higher recall and lower precision than computational linguistics methods (Jenssen et al., 2001). Although statistical linguistics had been successfully used in microarray analysis (Ganjendran et al., 2007; Hsu et al., 2007; Li et al., 2007), a combination of statistical and computational linguistics had not been used.

Prior to this study, developers of biomedical literature analysis systems had not considered the use of a completely generic text processing tool to process biomedical text for the purpose of extracting protein-protein interactions although generic text processing tools had been used in other fields of study (Grover et al., 2002; van Eck and van den Berg, 2005). This study has demonstrated that Muscorian, using an un-adapted generic text processing tool, is capable of comparable performance in extracting

protein-protein interactions from published literature (Chapter 2) and Chapter 3 had argued that poor parts-of-speech (POS) tagging performance might be less critical to the overall performance of the text analysis system due to alternative POS tag use by the shallow parser, resulting in a large number of POS tagging errors being absolved. This suggests that error rate at each stage of natural language processing may not add to the error rate of the entire natural language processing system. Hence, it seems appropriate to evaluate the performance of the entire text processing system rather than its individual components.

However, the limitation of Muscorian is the dependency on an entity normalization process which converts full-length protein names into abbreviations. This requires a priori knowledge of the protein names and assumes that 2 different proteins are not shortened to the same abbreviated name. In this study, confusion (different protein names with the same abbreviation) and errors with protein name abbreviations are reduced with an additional step of manual curation of the abbreviation dictionary. Protein name recognition (gene mention) and normalization of protein names to a unique identifier such as Unigene ID (gene normalization) is likely to be more optimal than the current method used in this study. Gene mention and normalization are topics of challenges in the latest BioCreative workshop in 2006. Hence, with improved techniques for gene name identification and normalizing to unique identifiers, protein-protein interactions extraction can be more robust.

Chapter 4 demonstrated that about 99% of the protein-protein interactions extracted by computation linguistics method can be mapped onto 1-mention PubGene co-occurrence network. This suggests that protein-protein interactions extracted by computation linguistics method can be used to annotate and supplement statistical linguistics information extraction. Although this is a proof-of-concept that computation linguistics can be used to annotate statistical co-occurrence network, it can be argued that using only 2 types of interactions is limiting in scope. Hence, future work on Muscorian can expand to extract all 24 interactions described by Textpresso (Muller et al., 2004).

The flexibility of Muscorian lies in the generalization of text into subject-verb-object structure and extracting specific assertions from this structure. This

suggests that Muscorian can extract assertions if these assertions could be captured in the form of subject-verb-object. Hence, it may be possible to use Muscorian extract assertions on protein localization (for example, “Protein X” “accumulate” “plasma membrane”) which could then be evaluated using the cellular component subset of Gene Ontology. The subject-verb-object structure can be explained as two nouns linked by a verb. The extracted nouns might be used in a thematic analysis of a collection of literature (Wilbur, 2002) and the relationships between themes. In addition, by searching for the name of a disease in one of the 2 nouns and a protein in another, Muscorian may be used to extract assertions on the proteins associated to a disease. However, the above-mentioned uses requires further evaluation.

#### ***6.4 Linguistic Analysis can Filter Microarray Data for Potentially Novel Hypotheses***

Microarray can be used to screen for biologically important transcriptional changes following treatment or experimental manipulation and Reverter et al. (2005) has shown that genes with co-expression profiles are more functionally correlated than functionally unrelated genes. However, it is not possible to know which co-expressed pairs of genes are suitable areas for future work without a priori analysis of the literature. The work in Chapter 4 highlights the use of high recall text analysis to represent a sieve of current scientific knowledge to filter co-expressed pairs of genes for those un-described in the literature, representing potentially novel hypotheses for future work. Nevertheless, the question remains – how should these hypotheses be evaluated (Cohen and Hersh, 2005)? Faced with a contradictory scenario of evaluating untested hypotheses, several means of scoring them based on available information had been attempted (Jelier et al., 2008; Smalheiser et al., 2009). Chapter 4 ranks the list of potentially novel hypotheses by their degree of correlation using Pearson's correlation coefficient and found 7 pairs of genes with near perfect correlation which were not found in the literature analysis, suggesting that this may be useful to narrow a large gene list from microarray analysis to a small set of genes for further examination.

It seems plausible to employ this technique for the analysis of other phases of mouse

lactation such as the study of pathways in involution (Brennan et al., 2008) or other microarray data sets. NCBI's Gene Expression Omnibus and Stanford Microarray Database has a combined public collection of more than 18 thousand large-scale expression data sets comprising of more than half a million biological samples. This represents a large untapped resource of experimental data from which many novel hypotheses can emerge. This study (Chapter 4) had also demonstrated that the use of multiple comparative data sets for analysis can be a very powerful technique to yield high quality hypotheses (Margaret Neville, Personal Communication) as the number of co-expressed gene-pairs are reduced to 1140 (co-expressed gene-pairs found in 4 data sets) from 210283 (co-expressed gene-pairs found in 1 data set), or a 99.46% reduction.

In addition to the use of text-mined interactions to filter microarray correlated gene-pairs for potential novel hypotheses for future research, it seem plausible that the reverse may be used – using microarray co-expression statistics as a measure of confidence in text-mined interactions.

Despite success in identifying a small set of potentially novel hypotheses in this study, there are 2 limitations in the current study. Firstly, this study focused on mouse which accounts for about 5 percent of all articles indexed PubMed. On the other hand, human-related studies account for about half of the research articles indexed in PubMed. Hence, it will be useful to expand the methods presented in this thesis to human studies or even the entire PubMed as attempted by Jenssen et al. (2001) as using a larger portion of PubMed may reduce false positive. Secondly, the construction of gene interaction (co-expression) network from microarray data is based on Pearson's coefficient of correlation. Although the strength of this method is simplicity and needed minimal computing power, it is likely to result in higher false positive rate than stronger and more computational intensive statistical methods such as Bayesian network (Chen et al., 2006) but the computing power needed for these methods to process entire sets of microarray data is usually prohibitive. However, correlation-based network can be used to reduce computational complexity by reducing the number of interactions to calculate as further computation is needed co-expressed genes and not all pair-wise possibilities. Therefore, further work can be aimed at reducing false positive rate by either using stronger statistical methods locally on subsets of the original data.



Despite the limitations as described above, Muscorian has a pre-defined list of proteins and interaction database which may be useful for other researchers to examine possible interactions to their protein of interest. Therefore, it is feasible to evaluate Muscorian in terms of usability by other biomedical researchers to solve real biological problems (Cohen and Hersh, 2005).

## **6.5 *Changing Face of Biological Research***

Recent advances in genomics, transcriptomics and proteomics experimental technology allowed for high-throughput screening to be carried out, giving rise to large amount of experimental data. Taking transcriptomics as an example, microarray technology can examine the expression of more than 12 thousand genes in a single experiment compared to Northern hybridization which can only examine the expression of less than 10 genes per experiment 2 decades ago. These technological advances in high-throughput experimental techniques had led to a revolution in the philosophy and method of doing biological research, effectively separating the philosophy and method of doing biological research into pre-genomics era and post-genomics era.

The scientific method of biological research in the pre-genomics era is a recursive 4-step linear process. Firstly, the problem, null and alternate hypotheses are formulated with support from the literature. Secondly, the actual experiments are carried out in attempt to disprove the null hypothesis. Thirdly, data is collected from the experiments. Finally, the data collected is analyzed in an attempt to disprove the null hypothesis. The analysis is then used to formulate the next problem and its hypotheses.

The explosion of experimental data as a result of technological advances had led to a 6-step non-linear scientific method of biological research in the post-genomic era. Firstly, the problem, null and alternate hypotheses are formulated with support from the literature. Secondly, a survey for suitable data sets originated from related studies is carrying out before performing the intended experiments. For example, this research used data sets from Master et al. (2002) and Clarkson and Watson (2003). Thirdly, additional experiments are designed in an attempt to disprove the null hypothesis. Fourthly, data is collected from the experiments. Fifthly, maintaining a continued survey

of new data sets from related studies while Step 3 and 4 were being carried out and modifying additional experiments to maximize the use of these new data sets. In this study, new microarray data sets from Stein et al. (2004) and Rudolph et al. (2007) were made available. Finally, the data collected is analyzed in attempt to disprove the null hypothesis. The analysis is then used to formulate the next problem and its hypotheses. This 6-step non-linear scientific method of biological research in the post-genomic era can lead to maximum utilization of research funding and greater collaboration.

## **6.6 Summary**

This thesis demonstrated that the IFP-induced mouse mammary explant does not mimic secretory differentiation at the transcription level. However, IFP-induced mouse mammary explant may provide a good basis to elucidate other hormones needed for secretory differentiation.

Previous assumption that generic computational linguistics processor is unable to process biomedical text due to domain-specificity is not supported in the current study and is attributed to complementary parts-of-speech tag use in the shallow parsing (breaking down sentences into phrases) process.

Subject-verb-object structure has been shown to be a suitable intermediate for extracting protein-protein interactions from text and information extracted from subject-verb-object structure can supplement information extraction by statistical co-occurrence.

Using computational and statistical information extraction, a filter representing the current state of biological knowledge can be used to filter microarray analysis for the identification of potential novel hypotheses for further research.

## References

- ABELL, K., BILANCIO, A., CLARKSON, R. W., TIFFEN, P. G., ALTAPARMAKOV, A. I., BURDON, T. G., ASANO, T., VANHAESEBROECK, B. & WATSON, C. J. (2005) Stat3-induced apoptosis requires a molecular switch in PI(3)K subunit composition. *Nature Cell Biology*, 7, 392-8.
- ABULAISH, M. & DEY, L. (2007) Biological relation extraction and query answering from MEDLINE abstracts using ontology-based text mining. *Data & Knowledge Engineering*, 61, 228.
- ACCORSI, P. A., PACIONI, B., PEZZI, C., FORNI, M., FLINT, D. J. & SEREN, E. (2002) Role of prolactin, growth hormone, and insulin-like growth factor 1 in mammary gland involution in the dairy cow. *Journal of Dairy Science*, 85, 507-513.
- ADAR, E. (2004) SaRAD: a Simple and Robust Abbreviation Dictionary. *Bioinformatics*, 20, 527-33.
- AERTS, S., HAEUSSLER, M., VAN VOOREN, S., GRIFFITH, O. L., HULPIAU, P., JONES, S. J., MONTGOMERY, S. B. & BERGMAN, C. M. (2008) Text-mining assisted regulatory annotation. *Genome Biology*, 9, R31.
- AGGELER, J., PARK, C. S. & BISSELL, M. J. (1988) Regulation of milk protein and basement membrane gene expression: the influence of the extracellular matrix. *Journal of Dairy Science*, 71, 2830-42.
- AHONEN, T. J., HARKONEN, P. L., RUI, H. & NEVALEINEN, M. T. (2002) PRL Signal Transduction in the Epithelial Compartment of Rat Prostate Maintained as Long-Term Organ Cultures in Vitro. *Endocrinology*, 143, 228-238.
- ALAKO, B. T., VELDHoven, A., VAN BAAL, S., JELIER, R., VERHOEVEN, S., RULLMANN, T., POLMAN, J. & JENSTER, G. (2005) CoPub Mapper: mining MEDLINE based on search term co-publication. *BMC Bioinformatics*, 6, 51.
- ALEX, B., GROVER, C., HADDOW, B., KABADJOV, M., KLEIN, E., MATTHEWS, M., ROEBUCK, S., TOBIN, R. & WANG, X. (2008) Assisted curation: does text mining really help? *Pacific Symposium on Biocomputing*, 556-67.
- ALFARANO, C., ANDRADE, C. E., ANTHONY, K., BAHROOS, N., BAJEC, M., BANTOFT, K., BETEL, D., BOBECHKO, B., BOUTILIER, K., BURGESS, E., BUZADZIJA, K., CAVERO, R., D'ABREO, C., DONALDSON, I., DORAIRAJOO, D., DUMONTIER, M. J., DUMONTIER, M. R., EARLES, V., FARRALL, R., FELDMAN, H., GARDERMAN, E., GONG, Y., GONZAGA, R., GRYTSAN, V., GRYZ, E., GU, V., HALDORSEN, E., HALUPA, A., HAW,

R., HRVOJIC, A., HURRELL, L., ISSERLIN, R., JACK, F., JUMA, F., KHAN, A., KON, T., KONOPINSKY, S., LE, V., LEE, E., LING, S., MAGIDIN, M., MONIAKIS, J., MONTOJO, J., MOORE, S., MUSKAT, B., NG, I., PARAISO, J. P., PARKER, B., PINTILIE, G., PIRONE, R., SALAMA, J. J., SGRO, S., SHAN, T., SHU, Y., SIEW, J., SKINNER, D., SNYDER, K., STASIUK, R., STRUMPF, D., TUEKAM, B., TAO, S., WANG, Z., WHITE, M., WILLIS, R., WOLTING, C., WONG, S., WRONG, A., XIN, C., YAO, R., YATES, B., ZHANG, S., ZHENG, K., PAWSON, T., OUELLETTE, B. F. & HOGUE, C. W. (2005) The Biomolecular Interaction Network Database and related tools 2005 update. *Nucleic Acids Research*, 33, D418-24.

ALLEN, J. (1994) *Natural language understanding*, New York, Benjamin-Cummings Publishing Company.

ALTMAN, R. B., BERGMAN, C. M., BLAKE, J., BLASCHKE, C., COHEN, A., GANNON, F., GRIVELL, L., HAHN, U., HERSH, W., HIRSCHMAN, L., JENSEN, L. J., KRALLINGER, M., MONS, B., O'DONOGHUE, S. I., PEITSCH, M. C., REBHOLZ-SCHUHMANN, D., SHATKAY, H. & VALENCIA, A. (2008) Text mining for biology--the way forward: opinions from leading scientists. *Genome Biology*, 9 Suppl 2, S7.

AMRANI, A., ROCHE, M., KODRATOFF, Y. & MATTE-TAILLIEZ, O. (2005) Inductive Improvement of Part-of-Speech Tagging and Its Effect on a Terminology of Molecular Biology. *18th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2005*. Victoria, Canada, Springer Berlin / Heidelberg.

ANDERSON, S. M., RUDOLPH, M. C., MCMANAMAN, J. L. & NEVILLE, M. C. (2007) Key stages in mammary gland development. Secretory activation in the mammary gland: it's not just about milk protein synthesis! *Breast Cancer Research*, 9, 204.

AOKI, N., KAWAMURA, M., YAMAGUCHI-AOKI, Y., OHIRA, S. & MATSUDA, T. (1999) Down-regulation of protein tyrosine phosphatase gene expression in lactating mouse mammary gland. *Journal of Biochemistry (Tokyo)*, 125, 669-75.

AOKI, N. & MATSUDA, T. (2000) A cytosolic protein-tyrosine phosphatase PTP1B specifically dephosphorylates and deactivates prolactin-activated STAT5a and STAT5b. *Journal of Biological Chemistry*, 275, 39718-26.

AOKI, N. & MATSUDA, T. (2002) A nuclear protein tyrosine phosphatase TC-PTP is a potential negative regulator of the PRL-mediated signaling pathway: dephosphorylation and deactivation of signal transducer and activator of transcription 5a and 5b by TC-PTP in nucleus. *Molecular Endocrinology*, 16, 58-69.

ARONSON, A. R. (2001) Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. *Proceedings of the AMIA Annual*

*Symposium*, 17-21.

ARONSON, A. R., BODENREIDER, O., CHANG, H. F., HUMPHREY, S. M., MORK, J. G., NELSON, S. J., RINDFLESCH, T. C. & WILBUR, W. J. (2000) The NLM Indexing Initiative. *Proceedings of the AMIA Annual Symposium*, 17-21.

ARONSON, A. R. & RINDFLESCH, T. C. (1997) Query expansion using the UMLS Metathesaurus. *Proceedings of the AMIA Annual Symposium*, 485-9.

ASHBURNER, M., BALL, C. A., BLAKE, J. A., BOTSTEIN, D., BUTLER, H., CHERRY, J. M., DAVIS, A. P., DOLINSKI, K., DWIGHT, S. S., EPPIG, J. T., HARRIS, M. A., HILL, D. P., ISSEL-TARVER, L., KASARSKIS, A., LEWIS, S., MATESE, J. C., RICHARDSON, J. E., RINGWALD, M., RUBIN, G. M. & SHERLOCK, G. (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics*, 25, 25-9.

BADER, G. D. & HOGUE, C. W. (2003) An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4, 2.

BAIK, M. G., LEE, M. J. & CHOI, Y. J. (1998) Gene expression during involution of mammary gland (review). *International Journal of Molecular Medicine*, 2, 39-44.

BAKER, C. J., KANAGASABAI, R., ANG, W. T., VEERAMANI, A., LOW, H. S. & WENK, M. R. (2008) Towards ontology-driven navigation of the lipid bibliosphere. *BMC Bioinformatics*, 9 Suppl 1, S5.

BALDI, P. & LONG, A. D. (2001) A Bayesian framework for the analysis of microarray expression data: regularized t-test and statistical inference of gene changes. *Bioinformatics*, 17.

BALL, S. M. (1998) The development of the terminal end bud in the prepubertal-pubertal mouse mammary gland. *Anatomical Records*, 250, 459-64.

BAMBERGER, C. M., SCHULTE, H. M. & CHROUSOS, G. P. (1996) Molecular determinants of glucocorticoid receptor function and tissue sensitivity to glucocorticoids. *Endocrine Review*, 17, 245-61.

BARASH, I., FAERMAN, A., PUZIS, R., PETERSON, D. & SHANI, M. (1995) Synthesis and secretion of caseins by the mouse mammary gland: production and characterization of new polyclonal antibodies. *Molecular and Cellular Biochemistry*, 144, 175-80.

BARNAWELL, E. B. (1965) A comparative study of the responses of mammary tissues from several mammalian species to hormones in vitro. *Journal of Experimental Zoology*, 160, 189-206.

- BARRETT, T. & EDGAR, R. (2006) Gene expression omnibus: microarray data storage, submission, retrieval, and analysis. *Methods in Enzymology*, 411, 352-69.
- BARSKY, A., GARDY, J. L., HANCOCK, R. E. & MUNZNER, T. (2007) Cerebral: a Cytoscape plugin for layout of and interaction with biological networks using subcellular localization annotation. *Bioinformatics*, 23, 1040-2.
- BASU, S., BREMER, E., ZHOU, C. & BOGENHAGEN, D. F. (2006) MiGenes: a searchable interspecies database of mitochondrial proteins curated using gene ontology annotation. *Bioinformatics*, 22, 485-92.
- BAUMGARTNER, W. A., JR., LU, Z., JOHNSON, H. L., CAPORASO, J. G., PAQUETTE, J., LINDEMANN, A., WHITE, E. K., MEDVEDEVA, O., COHEN, K. B. & HUNTER, L. (2008) Concept recognition for extracting protein interaction relations from biomedical text. *Genome Biology*, 9 Suppl 2, S9.
- BECKER, K. G., HOSACK, D. A., DENNIS, G., JR., LEMPICKI, R. A., BRIGHT, T. J., CHEADLE, C. & ENGEL, J. (2003) PubMatrix: a tool for multiplex literature mining. *BMC Bioinformatics*, 4, 61.
- BEISSBARTH, T. (2006) Interpreting experimental results using gene ontologies. *Methods in Enzymology*, 411, 340-52.
- BEKHUIS, T. (2006) Conceptual biology, hypothesis discovery, and text mining: Swanson's legacy. *Biomedical Digital Libraries*, 3, 2.
- BELDADE, P., RUDD, S., GRUBER, J. D. & LONG, A. D. (2006) A wing expressed sequence tag resource for *Bicyclus anynana* butterflies, an evo-devo model. *BMC Genomics*, 7, 130.
- BEQUETTE, B. J., KYLE, C. E., CROMPTON, L. A., ANDERSON, S. E. & HANIGAN, M. D. (2002) Protein metabolism in lactating goats subjected to the insulin clamp. *Journal of Dairy Science*, 85, 1546-1555.
- BEQUETTE, B. J., KYLE, C. E., CROMPTON, L. A., BUCHAN, V. & HANIGAN, M. D. (2001) Insulin regulates milk production and mammary gland and hind-leg amino acid fluxes and blood flow in lactating goats. *Journal of Dairy Science*, 84, 241-255.
- BERLATO, C. & DOPPLER, W. (2009) Selective response to insulin versus IGF-I and IGF-II and upregulation of insulin-receptor splice variant B in the differentiated mouse mammary epithelium. *Endocrinology*.
- BEUMING, T., SKRABANEK, L., NIV, M. Y., MUKHERJEE, P. & WEINSTEIN, H. (2005) PDZBase: a protein-protein interaction database for PDZ-domains. *Bioinformatics*, 21, 827-8.

- BHATTACHARJEE, M., BOTTING, C. H. & SILLANPAA, M. J. (2008) Bayesian biomarker identification based on marker-expression proteomics data. *Genomics*, 92, 384-92.
- BIAGINI, R. E., KRIEG, E. F., PINKERTON, L. E. & HAMILTON, R. G. (2001) Receiver operating characteristics analyses of Food and Drug Administration-cleared serological assays for natural rubber latex-specific immunoglobulin E antibody. *Clinical and Diagnostic Laboratory Immunology*, 8, 1145-9.
- BIANCO, C., ADKINS, H. B., WECHSELBERGER, C., SENO, M., NORMANNO, N., DE LUCA, A., SUN, Y., KHAN, N., KENNEY, N., EBERT, A., WILLIAMS, K. P., SANICOLA, M. & SALOMON, D. S. (2002) Cripto-1 activates nodal- and ALK4-dependent and -independent signaling pathways in mammary epithelial Cells. *Molecular and Cellular Biology*, 22, 2586-97.
- BICKEL, D. R., MONTAZERI, Z., HSIEH, P. C., BEATTY, M., LAWIT, S. J. & BATE, N. J. (2009) Gene network reconstruction from transcriptional dynamics under kinetic model uncertainty: A case for the second derivative. *Bioinformatics*, 25, 772-9.
- BIDAUT, G. & STOECKERT, C. J., JR. (2009) Characterization of unknown adult stem cell samples by large scale data integration and artificial neural networks. *Pacific Symposium on Biocomputing*, 356-67.
- BLASCHKE, C., ANDRADE, M. A., OUZOUNIS, C. & VALENCIA, A. (1999) Automatic extraction of biological information from scientific text: protein-protein interactions. *Proceedings of the International Conference for Intelligent Systems in Molecular Biology*, 60-7.
- BOBY, T., PATCH, A. M. & AVES, S. J. (2005) TRbase: a database relating tandem repeats to disease genes for the human genome. *Bioinformatics*, 21, 811-6.
- BODENREIDER, O. & MITCHELL, J. A. (2003) Graphical visualization and navigation of genetic disease information. *AMIA Annual Symposium*.
- BOHNET, H. G., GOMEZ, F. & FRIESEN, H. G. (1977) Prolactin and estrogen binding sites in the mammary gland of lactating and non-lactating rat. *Endocrinology*, 101, 1111-1121.
- BOLANDER, F. F., JR. (1983) Persistent alterations in hormonal sensitivities of mammary glands from parous mice. *Endocrinology*, 112, 1796-800.
- BOLANDER, F. F., JR. (1984) Enhanced endocrine sensitivity in mouse mammary glands: hormonal requirements for induction and maintenance. *Endocrinology*, 115, 630-3.
- BOLANDER, F. F. J., NICHOLAS, K. R. & TOPPER, Y. J. (1979) Retention of

glucocorticoid by isolated mammary tissue may complicate interpretation of results from *in vitro* experiments. *Biochemical Biophysical Research Communication*, 91, 247-252.

- BOLANDER, F. F. J., NICHOLAS, K. R., WYK, J. J. V. & TOPPER, Y. J. (1981) Insulin is essential for accumulation of casein mRNA in mouse mammary epithelial cells. *Proceedings of the National Academy of Science*, 78, 5682-5684.
- BOLE-FEYSOT, C., GOFFIN, V., EDERY, M., BINART, N. & KELLY, P. A. (1998) Prolactin (PRL) and its receptor: actions, signal transduction pathways and phenotypes observed in PRL receptor knockout mice. *Endocrine Review*, 19, 225-68.
- BOUTROS, P. C. & OKEY, A. B. (2005) Unsupervised pattern recognition: an introduction to the whys and wherefores of clustering microarray data. *Briefings in Bioinformatics*, 6, 331-43.
- BRADY, S. M., LONG, T. A. & BENFEY, P. N. (2006) Unraveling the dynamic transcriptome. *Plant Cell*, 18, 2101-11.
- BRANTS, T. (2000) TnT - a statistical part-of-speech tagger. *6th Applied Natural Language Processing Conference*. Seattle, Washington, USA.
- BRENNAN, A. J., SHARP, J. A., KHALIL, E., DIGBY, M. R., MAILER, S. L., LEFEVRE, C. M. & NICHOLAS, K. R. (2008) A population of mammary epithelial cells do not require hormones or growth factors to survive. *Journal of Endocrinology*, 196, 483-496.
- BRILL, E. (1995) Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. *Computational Linguistics*, 21, 543-565.
- BRISKEN, C., KAUR, S., CHAVARRIA, T. E., BINART, N., SUTHERLAND, R. L., WEINBERG, R. A., KELLY, P. A. & ORMANDY, C. J. (1999) Prolactin controls mammary gland development via direct and indirect mechanisms. *Developmental Biology*, 210, 96-106.
- BROBERG, P. (2003) Statistical methods for ranking differentially expressed genes. *Genome Biology*, 4, R41.
- BROCKMAN, J. M., SINGH, P., LIU, D., QUINLAN, S., SALISBURY, J. & GRABER, J. H. (2005) PACdb: PolyA Cleavage Site and 3'-UTR Database. *Bioinformatics*, 21, 3691-3.
- BROWN, C. S., GOODWIN, P. C. & SORGER, P. K. (2001) Image metrics in the statistical analysis of DNA microarray data. *Proceedings of the National Academy of Science U S A*, 98, 8944-8949.



- BROWN, M. P. S., GRUNDY, W. N., LIN, D., SUGNET, C., M. ARES, J. & HAUSSLER, D. (2000) Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Science U S A*, 97, 262-267.
- BURNOL, A. F., LETURQUE, A., FERRE, P., KANDE, J. & GIRARD, J. (1986) Increased insulin sensitivity and responsiveness during lactation in rats. *American Journal of Physiology*, 251, E537-41.
- BUSSMANN, L. E., WARD, S. & KUHN, N. J. (1984) Lactose and fatty acid synthesis in lactating-rat mammary gland. Effects of starvation, re-feeding, and administration of insulin, adrenaline, streptozotocin and 2-bromo-alpha-ergocryptine. *Biochemical Journal*, 219, 173-80.
- BUTLER, S. T., MARR, A. L., PELTON, S. H., RADCLIFF, R. P., LUCY, M. C. & BUTLER, W. R. (2003) Insulin restores GH responsiveness during lactation-induced negative energy balance in dairy cattle: effects on expression of IGF-1 and GH receptor 1A. *Journal of Endocrinology*, 176, 205-217.
- BUTTE, A. J., YE, J., NIEDERFELLNER, G., RETT, K., HARING, H. U., WHITE, M. F. & KOHANE, I. S. (2001) Determining significant fold differences in gene expression analysis. *Proceedings of the Pacific Symposium of Biocomputing*, 6, 6-17.
- CAKMAK, A. & OZSOYOGLU, G. (2008) Discovering gene annotations in biomedical text databases. *BMC Bioinformatics*, 9, 143.
- CANO, C., MONAGHAN, T., BLANCO, A., WALL, D. P. & PESHKIN, L. (2009) Collaborative text-annotation resource for disease-centered relation extraction from biomedical text. *Journal of Biomedical Informatics*.
- CAPORASO, J. G., DESHPANDE, N., FINK, J. L., BOURNE, P. E., COHEN, K. B. & HUNTER, L. (2008) Intrinsic evaluation of text mining tools may not predict performance on realistic tasks. *Pacific Symposium on Biocomputing*, 640-51.
- CARRASCOSA, J. M., RAMOS, P., MOLERO, J. C. & HERRERA, E. (1998) Changes in the kinase activity of the insulin receptor account for an increased insulin sensitivity of mammary gland in late pregnancy. *Endocrinology*, 139, 520-6.
- CHANDRA, N. R., KUMAR, N., JEYAKANI, J., SINGH, D. D., GOWDA, S. B. & PRATHIMA, M. N. (2006) Lectindb: a plant lectin database. *Glycobiology*, 16, 938-46.
- CHANG, J. T. (2003) Using machine learning to extract drug and gene relationships from text. Stanford University.
- CHANG, J. T., SCHUTZE, H. & ALTMAN, R. B. (2002) Creating an online dictionary

of abbreviations from MEDLINE. *Journal of the American Medical Informatics Association*, 9, 612-620.

- CHANG, J. T., SCHUTZE, H. & ALTMAN, R. B. (2004) GAPSCORE: finding gene and protein names one word at a time. *Bioinformatics*, 20, 216-225.
- CHAPMAN, S., SCHENK, P., KAZAN, K. & MANNERS, J. (2002) Using biplots to interpret gene expression patterns in plants. *Bioinformatics*, 18, 202-204.
- CHATR-ARYAMONTRI, A., KERRIEN, S., KHADAKE, J., ORCHARD, S., CEOL, A., LICATA, L., CASTAGNOLI, L., COSTA, S., DEROW, C., HUNTLEY, R., ARANDA, B., LEROY, C., THORNEYCROFT, D., APWEILER, R., CESARENI, G. & HERMJAKOB, H. (2008) MINT and IntAct contribute to the Second BioCreative challenge: serving the text-mining community with high quality molecular interaction data. *Genome Biology*, 9 Suppl 2, S5.
- CHEN, E. S., HRIPCSAK, G., XU, H., MARKATOU, M. & FRIEDMAN, C. (2008) Automated acquisition of disease drug knowledge from biomedical and clinical documents: an initial study. *Journal of the American Medical Informatics Association*, 15, 87-98.
- CHEN, Y., KAMAT, V., DOUGHERTY, E. R., BITTNER, M. L., MELTZER, P. S. & TRENT, J. M. (2002) Ratio statistics of gene expression level and applications to microarray data analysis. *Bioinformatics*, 18, 1207-1215.
- CHEN, X., CHEN, M. & NING, K. (2006) BNArray: an R package for constructing gene regulatory networks from microarray data by using Bayesian network. *Bioinformatics*, 22, 2952-4.
- CHEN, E. S., HRIPCSAK, G., XU, H., MARKATOU, M. & FRIEDMAN, C. (2008) Automated Acquisition of Disease Drug Knowledge from Biomedical and Clinical Documents: An Initial Study. *Journal of the American Medical Informatics Association*, 15, 87-98.
- CHEN, D., MULLER, H. M. & STERNBERG, P. W. (2006b) Automatic document classification of biological literature. *BMC Bioinformatics*, 7, 370.
- CHEN, H. & SHARP, B. M. (2004) Content-rich biological network constructed by mining PubMed abstracts. *BMC Bioinformatics*, 5, 147.
- CHI, Y. Y., IBRAHIM, J. G., BISSAHOYO, A. & THREADGILL, D. W. (2007) Bayesian hierarchical modeling for time course microarray experiments. *Biometrics*, 63, 496-504.
- CHIANG, J.-H. & YU, H.-C. (2003) MeKE: discovering the functions of gene products from biomedical literature via sentence alignment. *Bioinformatics*, 19, 1417-1422.

- CHIANG, J.-H., YU, H.-C. & HSU, H.-J. (2004) GIS: a biomedical text-mining system for gene information discovery. *Bioinformatics*, 20, 120.
- CHIANG, J. H., SHIN, J. W., LIU, H. H. & CHIN, C. L. (2006) GeneLibrarian: an effective gene-information summarization and visualization system. *BMC Bioinformatics*, 7, 392.
- CHOMCZYNSKI, P., QASBA, P. & TOPPER, Y. J. (1984) Essential role of insulin in transcription of the rat 25,000 molecular weight casein gene. *Science*, 226, 1326-1328.
- CHOMCZYNSKI, P., QASBA, P. & TOPPER, Y. J. (1986) Transcriptional and post-transcriptional roles of glucocorticoid in the expression of the rat 25,000 molecular weight casein gene. *Biochemical Biophysical Research Communications*, 134, 812-818.
- CHURCHILL, G. A. (2004) Using ANOVA to analyze microarray data. *Biotechniques*, 37, 173-5, 177.
- CLARKSON, R. W. E. & WATSON, C. J. (2003) Microarray analysis of the involution switch. *Journal of Mammary Gland Biology and Neoplasia*, 8, 309-319.
- CLEVELAND, W. S. (1981) LOWESS: A program for smoothing scatterplots by robust locally weighted regression. *The American Statistician*, 35, 54.
- COBLITZ, B., WU, M., SHIKANO, S. & LI, M. (2006) C-terminal binding: an expanded repertoire and function of 14-3-3 proteins. *FEBS Letters*, 580, 1531-5.
- COHEN, A. M. & HERSH, W. R. (2005) A survey of current work in biomedical text mining. *Briefings in Bioinformatics*, 6, 57-71.
- COHEN, A. M., HERSH, W. R., DUBAY, C. & SPACKMAN, K. (2005) Using co-occurrence network structure to extract synonymous gene and protein names from MEDLINE abstracts. *BMC Bioinformatics*, 6, 103.
- COHEN, K. B. & HUNTER, L. (2008) Getting started in text mining. *PLoS Computational Biology*, 4, e20.
- COLLIER, R. J., BAUMAN, D. E. & HAYS, R. L. (1977) Lactogenesis in explant cultures of mammary tissue from pregnant cows. *Endocrinology*, 100, 1192-200.
- COLLIER, N., PARK, H. S., OGATA, N., TATEISHI, Y., NOBATA, C., OHTA, T., SEKIMIZU, T., IMAI, H., IBUSHI, K. & TSUJII, J.-I. (1999) The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers. *Ninth Conference of the European Chapter of the Association for Computational Linguistics*. Association of Computational Linguistics.

- COLOSIMO, M. E., MORGAN, A. A., YEH, A. S., COLOMBE, J. B. & HIRSCHMAN, L. (2005) Data preparation and interannotator agreement: BioCreAtIvE task 1B. *BMC Bioinformatics*, 6 Suppl 1, S12.
- COOPER, J. & BYRD, R. (1998) Lexical navigation: visually prompted query refinement. *ACM Digital Libraries Conference*. Philadelphia, PA, USA.
- COOPER, J. W. & KERSHENBAUM, A. (2005) Discovery of protein-protein interactions using a combination of linguistic, statistical and graphical information. *BMC Bioinformatics*, 6, 143.
- COULDREY, C., MOITRA, J., VINSON, C., ANVER, M., NAGASHIMA, K. & GREEN, J. (2002) Adipose tissue: a vital in vivo role in mammary gland development but not differentiation. *Dev Dyn*, 223, 459-68.
- COUTO, F. M., SILVA, M. J., LEE, V., DIMMER, E., CAMON, E., APWEILER, R., KIRSCH, H. & REBHOLZ-SCHUHMANN, D. (2006) GOAnnotator: linking protein GO annotations to evidence text. *Journal of Biomedical Discovery and Collaboration*, 1, 19.
- COVELL, D. G., WALLQVIST, A., RABOW, A. A. & THANKI, N. (2003) Molecular classification of cancer: unsupervised self-organizing map analysis of gene expression microarray data. *Molecular Cancer Therapeutics*, 2, 317-32.
- COWIE, A. T. & TINDAL, J. S. (1961) The maintenance of lactation in the rat after hypophysial anterior lobectomy during pregnancy. *Journal of Endocrinology*, 22, 403-8.
- COWIE, A. T., TINDAL, J. S. & BENSON, G. K. (1960) Pituitary grafts and milk secretion in hypophysectomized rats. *Journal of Endocrinology*, 21, 115-23.
- CRAVEN, M. & KUMLIEN, J. (1999) Constructing biological knowledge bases by extracting information from text sources. *Proceedings of the International Conference for Intelligent System in Molecular Biology*, 77-86.
- CRIM, J., MCDONALD, R. & PEREIRA, F. (2005) Automatically annotating documents with normalized gene lists. *BMC Bioinformatics*, 6 Suppl 1, S13.
- CRYSTAL, D. (1997) *The Cambridge Encyclopedia of Languages*, Cambridge, Cambridge University Press.
- CULHANE, A. C., PERRIERE, G., CONSIDINE, E. C., COTTER, T. G. & HIGGINS, D. G. (2002) Between-group analysis of microarray data. *Bioinformatics*, 18, 1600-1608.
- CUNNINGHAM, H. (2000) Software Architecture for Language Engineering. *Department of Computer Science*. University of Sheffield.

- CUSCHIERI, J. & MAIER, R. V. (2005) Mitogen-activated protein kinase (MAPK). *Critical Care in Medicine*, 33, S417-9.
- CUSSENS, J. & DZEROSKI, S. (Eds.) (2000) *Learning Languages in Logic*, Berlin, Heidelberg, New York, Barcelona, Hong Kong, London, Milan, Paris, Singapore, Tokyo, Springer.
- CUSSENS, J. & NÉDELLEC, C. (Eds.) (2005) *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, Bonn.
- DAELEMANS, W. (2006) A Mission for Computational Natural Language Learning. IN MÀRQUEZ, L. & KLEIN, D. (Eds.) *The Tenth Conference on Natural Language Learning (CoNLL-X)*. New York City, USA, Association for Computational Linguistics.
- DANIEL, M. M., HSINCHUN, C., HUA, S. & BYRON, B. M. (2004) Extracting gene pathway relations using a hybrid grammar: the Arizona Relation Parser. *Bioinformatics*, 20, 3370.
- DARASELIA, N., YURYEV, A., EGOROV, S., NOVICHKOVA, S., NIKITIN, A. & MAZO, I. (2004) Extracting human protein interactions from MEDLINE using a full-sentence parser. *Bioinformatics*, 20, 604-11.
- DATTA, S. & DATTA, S. (2006) Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. *BMC Bioinformatics*, 7, 397.
- DAVID, P. A. C., BERNARD, F. B., WILLIAM, B. L. & DAVID, T. J. (2004) BioRAT: extracting biological information from full-length papers. *Bioinformatics*, 20, 3206.
- DAVIS, S. R., FARR, V. C., COPEMAN, P. J., CARRUTHERS, V. R., KNIGHT, C. H. & STELWAGEN, K. (1998) Partitioning of milk accumulation between cisternal and alveolar compartments of the bovine udder: relationship to production loss during once daily milking. *Journal of Dairy Research*, 65, 1-8.
- DE BRUIJN, B. & MARTIN, J. (2002) Getting to the (c)ore of knowledge: mining biomedical literature. *International Journal of Medical Informatics*, 67, 7-18.
- DE LA CRUZ, E. M. (2001) Actin-binding proteins: an overview. *Results and problems in cell differentiation*, 32, 123-34.
- DE LEON, M., WELCHER, A. A., NAHIN, R. H., LIU, Y., RUDA, M. A., SHOOTER, E. M. & MOLINA, C. A. (1996) Fatty acid binding protein is induced in neurons of the dorsal root ganglia after peripheral nerve injury. *Journal of Neuroscience Research*, 44, 283-92.

- DERIKX, J. P., MATTHIJSEN, R. A., DE BRUINE, A. P., VAN BIJNEN, A. A., HEINEMAN, E., VAN DAM, R. M., DEJONG, C. H. & BUURMAN, W. A. (2008) Rapid reversal of human intestinal ischemia-reperfusion induced damage by shedding of injured enterocytes and reepithelialisation. *PLoS ONE*, 3, e3428.
- DERNATAS, E. & KOKKINAKIS, G. (1995) Automatic stochastic tagging of natural language texts. *Computational Linguistics*, 21, 137-163.
- DESJARDINS, C., PAAPE, M. J. & TUCKER, H. A. (1968) Contribution of pregnancy, fetuses, fetal placentas and deciduomas to mammary gland and uterine development. *Endocrinology*, 83, 907-10.
- DEVINOY, E., HOUEBINE, L. M. & DELOUIS, C. (1978) Role of prolactin and glucocorticoids in the expression of casein genes in rabbit mammary gland organ culture. Quantification of casein mRNA. *Biochemica et Biophysica Acta*, 517, 360-6.
- DIDIER, G., BREZELLE, P., REMY, E. & HENAUT, A. (2002) Gene-ANOVA - gene expression analysis of variance. *Bioinformatics*, 18, 490-491.
- DING, J., BERLEANT, B., NETTLETON, D. & WURTELE, E. (2002) Mining MEDLINE: abstracts, sentences, or phrases? *Proceedings of the Pacific Symposium on Biocomputing*, 7, 326-337.
- DING, J., VISWANATHAN, K., BERLEANT, D., HUGHES, L., WURTELE, E. S., ASHLOCK, D., DICKERSON, J. A., FULMER, A. & SCHNABLE, P. S. (2005) Using the biological taxonomy to access biological literature with PathBinderH. *Bioinformatics*, 21, 2560-2.
- DIVOLI, A. & ATTWOOD, T. K. (2005) BioIE: extracting informative sentences from the biomedical literature. *Bioinformatics*, 21, 2138-9.
- DJIANE, J., CLAUSER, H. & KELLY, P. A. (1979) Rapid down-regulation of prolactin receptors in mammary gland and liver. *Biochemica et Biophysica Acta*, 90, 1371-1378.
- DOMS, A. & SCHROEDER, M. (2005) GoPubMed: exploring PubMed with the Gene Ontology. *Nucleic Acids Research*, 33, W783-6.
- DONALDSON, I., MARTIN, J., DE BRUIJN, B., WOLTING, C., LAY, V., TUEKAM, B., ZHANG, S., BASKIN, B., BADER, G. D., MICHALICKOVA, K., PAWSON, T. & HOGUE, C. W. (2003) PreBIND and Textomy--mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics*, 4, 11.
- DONG, B., HUANG, C., LI, D. & ZHAO, F. Q. (2009) Oct-1 functions as a transactivator in the hormonal induction of beta-casein gene expression.

- DONG, B. & ZHAO, F. Q. (2007) Involvement of the ubiquitous Oct-1 transcription factor in hormonal induction of beta-casein gene expression. *Biochemical Journal*, 401, 57-64.
- DOU, Y., BAISNEE, P. F., POLLASTRI, G., PECOUT, Y., NOWICK, J. & BALDI, P. (2004) ICBS: a database of interactions between protein chains mediated by beta-sheet formation. *Bioinformatics*, 20, 2767-77.
- DYBKAER, K., IQBAL, J., ZHOU, G., GENG, H., XIAO, L., SCHMITZ, A., D'AMORE, F. & CHAN, W. C. (2007) Genome wide transcriptional analysis of resting and IL2 activated human natural killer cells: gene expression signatures indicative of novel molecular signaling pathways. *BMC Genomics*, 8, 230.
- EDWARDS, C. B. & GURLAND, J. (1961) A Class of Distributions Applicable to Accidents. *Journal of the American Statistical Association*, 56, 503-517.
- EFRON, B. & TIBSHIRANI, R. (1986) Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statistical Science*, 1, 54-75.
- EGEA, P. F., SHAN, S. O., NAPETSCHNIG, J., SAVAGE, D. F., WALTER, P. & STROUD, R. M. (2004) Substrate twinning activates the signal recognition particle and its receptor. *Nature*, 427, 215-21.
- EGOROV, S., YURYEV, A. & DARASELIA, N. (2004) A simple and practical dictionary-based approach for identification of proteins in Medline abstracts. *Journal of the American Medical Informatics Association*, 11, 174-8.
- ELIAS, J. J. (1957) Cultivation of adult mouse mammary gland in hormone-enriched synthetic medium. *Science*, 126, 842-3.
- ELIAS, J. J. & RIVERA, E. (1959) Comparison of the responses of normal, precancerous, and neoplastic mouse mammary tissues to hormones in vitro. *Cancer Research*, 19, 505-11.
- ESLICK, I. & LIU, H. (2005) Langutils: A Natural Language Toolkit for Common Lisp. *International Lisp Conference*. Stanford University, USA.
- FANG, Y. C., HUANG, H. C. & JUAN, H. F. (2008) MeInfoText: associated gene methylation and cancer information from text mining. *BMC Bioinformatics*, 9, 22.
- FAULKIN, L. J. J. & DEOME, K. B. (1960) Regulation of growth and spacing of gland elements in the mammary fat pad of C3H mouse. *Journal of the National Cancer Institute*, 24, 953-969.

- FENG, C., YAMASHITA, F. & HASHIDA, M. (2007) Automated extraction of information from the literature on chemical-CYP3A4 interactions. *Journal of Chemical Information and Modeling*, 47, 2449-55.
- FERAL, C. C., NISHIYA, N., FENCZIK, C. A., STUHLMANN, H., SLEPAK, M. & GINSBERG, M. H. (2005) CD98hc (SLC3A2) mediates integrin signaling. *Proceedings of the National Academy of Science U S A*, 102, 355-60.
- FERRO, A., GIUGNO, R., PIGOLA, G., PULVIRENTI, A., SKRIPIN, D., BADER, G. D. & SHASHA, D. (2007) NetMatch: a Cytoscape Plugin for Searching Biological Networks. *Bioinformatics*.
- FINKEL, J., DINGARE, S., MANNING, C. D., NISSIM, M., ALEX, B. & GROVER, C. (2005) Exploring the boundaries: gene and protein identification in biomedical text. *BMC Bioinformatics*, 6 Suppl 1, S5.
- FLORENCE, H., ANTHONY, L. L. & FRED, E. C. (2004) Automated extraction of mutation data from the literature: application of MuteXt to G protein-coupled receptors and nuclear hormone receptors. *Bioinformatics*, 20, 557.
- FORSYTH, I. A. & TURVEY, A. (1984) Fatty acid synthesis by explant cultures from the mammary glands of goats on days 60 and 120 of pregnancy. *Journal of Endocrinology*, 100, 87-92.
- FRANKEL, L. B., CHRISTOFFERSEN, N. R., JACOBSEN, A., LINDOW, M., KROGH, A. & LUND, A. H. (2008) Programmed cell death 4 (PDCD4) is an important functional target of the microRNA miR-21 in breast cancer cells. *Journal of Biological Chemistry*, 283, 1026-33.
- FRANZEN, K., ERIKSSON, G., OLSSON, F., ASKER, L., LIDEN, P. & COSTER, J. (2002) Protein names and how to find them. *International Journal Medical Informatics*, 67, 49-61.
- FRIEDMAN, C. (2000) A broad-coverage natural language processing system. *AMIA Symposium*.
- FRIEDMAN, C., ALDERSON, P. O., AUSTIN, J. H., CIMINO, J. J. & JOHNSON, S. B. (1994) A general natural-language text processor for clinical radiology. *Journal of the American Medical Informatics Association*, 1, 161-74.
- FRIEDMAN, C., KRA, P., YU, H., KRAUTHAMMER, M. & RZHETSKY, A. (2001) GENIES: a natural-language processing system for the extraction of molecular pathways from journal articles. *Bioinformatics*, 17, S74-S82.
- FRISTENSKY, B. (2007) BIRCH: a user-oriented, locally-customizable, bioinformatics system. *BMC Bioinformatics*, 8, 54.



- FUHRMAN, S., CUNNINGHAM, M. J., WEN, X., ZWEIGER, G., SEIHAMER, J. J. & SOMOGYI, R. (2000) The application of Shannon entropy in the identification of putative drug targets. *Biosystem*, 55, 5-15.
- FUKUDA, K., TAMURA, A., TSUNODA, T. & TAKAGI, T. (1998) Toward information extraction: identifying protein names from biological papers. *Pacific Symposium on Biocomputing*, 707-18.
- FULKERSON, W. J., MCDOWELL, G. H. & FELL, L. R. (1975) Artificial induction of lactation in ewes: the role of prolactin. *Australian Journal of Biological Science*, 28, 525-30.
- FUNDEL, K., GUTTLER, D., ZIMMER, R. & APOSTOLAKIS, J. (2005) A simple approach for protein name identification: prospects and limits. *BMC Bioinformatics*, 6 Suppl 1, S15.
- GAIZAUSKAS, R., DEMETRIOU, G., ARTYMIUK, P. J. & WILLETT, P. (2003) Protein structures and information extraction from biological texts: the PASTA system. *Bioinformatics*, 19, 135-43.
- GAJENDRAN, V. K., LIN, J. R. & FYHRIE, D. P. (2007) An application of bioinformatics and text mining to the discovery of novel genes related to bone biology. *Bone*, 40, 1378-1388.
- GALLO, S. M., LI, L., HU, Z. & HALFON, M. S. (2006) REDfly: a Regulatory Element Database for Drosophila. *Bioinformatics*, 22, 381-3.
- GANSNER, E. R. & NORTH, S. C. (1999) An open graph visualization system and its applications to software engineering. *Software Practice and Experience*, 00, 1-5.
- GANGULY, R., GANGULY, N., MEHTA, N. M. & BANERJEE, M. R. (1980) Absolute requirement of glucocorticoid for expression of the casein gene in the presence of prolactin. *Proceedings of the National Academy of Science U S A*, 77, 6003-6.
- GANGULY, R., MEHTA, N. M., GANGULY, N. & BANERJEE, M. R. (1979) Glucocorticoid modulation of casein gene transcription in mouse mammary gland. *Proceedings of the National Academy of Science U S A*, 76, 6466-6470.
- GAO, G., ZHONG, Y., GUO, A., ZHU, Q., TANG, W., ZHENG, W., GU, X., WEI, L. & LUO, J. (2006) DRTF: a database of rice transcription factors. *Bioinformatics*, 22, 1286-7.
- GARCIA CASTRO, A., CHEN, Y. P. & RAGAN, M. A. (2005) Information integration in molecular bioscience. *Applied Bioinformatics*, 4, 157-73.
- GARCIA, O., SAVEANU, C., CLINE, M., FROMONT-RACINE, M., JACQUIER, A.,

- SCHWIKOWSKI, B. & AITTOKALLIO, T. (2007) Golorize: a Cytoscape plug-in for network visualization with Gene Ontology-based layout and coloring. *Bioinformatics*, 23, 394-6.
- GARDNER, A. M., VAILLANCOURT, R. R. & JOHNSON, G. L. (1993) Activation of mitogen-activated protein kinase/extracellular signal-regulated kinase kinase by G protein and tyrosine kinase oncoproteins. *Journal of Biological Chemistry*, 268, 17896-901.
- GARDNER, T. S., DI BERNARDO, D., LORENZ, D. & COLLINS, J. J. (2003) Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301, 102-5.
- GAUTHIER, J. P., LEGEAI, F., ZASADZINSKI, A., RISPE, C. & TAGU, D. (2007) AphidBase: a database for aphid genomic resources. *Bioinformatics*, 23, 783-4.
- GEORGE, R. A., SPRIGGS, R. V., THORNTON, J. M., AL-LAZIKANI, B. & SWINDELLS, M. B. (2004) SCOPEC: a database of protein catalytic domains. *Bioinformatics*, 20 Suppl 1, I130-I136.
- GERARD, S., EDWARD, A. F. & HARRY, W. (1983) Extended Boolean information retrieval. *Communications of the ACM*, 26, 1022-1036.
- GJENGSTO, P., PAUS, E., HALVORSEN, O. J., EIDE, J., AKSLEN, L. A., WENTZEL-LARSEN, T. & HOISAETER, P. A. (2005) Predictors of prostate cancer evaluated by receiver operating characteristics partial area index: a prospective institutional study. *Journal of Urology*, 173, 425-8.
- GOLL, J., RAJAGOPALA, S. V., SHIAU, S. C., WU, H., LAMB, B. T. & UETZ, P. (2008) MPIDB: the microbial protein interaction database. *Bioinformatics*, 24, 1743-4.
- GRIINARI, J. M., MCGUIRE, M. A., DWYER, D. A., BAUMAN, D. E., BARBANO, D. M. & HOUSE, W. A. (1997) The role of insulin in the regulation of milk protein synthesis in dairy cows. *Journal of Dairy Science*, 80, 2361-2371.
- GROMADA, J., BROCK, B., SCHMITZ, O. & RORSMAN, P. (2004) Glucagon-like peptide-1: regulation of insulin secretion and therapeutic potential. *Basic Clinical Pharmacology and Toxicology*, 95, 252-62.
- GRONER, B. (2002) Transcription factor regulation in mammary epithelial cells. *Domestic Animal Endocrinology*, 23, 25-32.
- GROVER, C., KLEIN, E., LASCARIDES, A. & LAPATA, M. (2002) XML-based NLP Tools for Analysing and Annotating Medical Language. *Second International Workshop on NLP and XML (NLPXML-2002)*. Taipei, Taiwan.

- GROVER, C., LAPATA, M. & LASCARIDES, A. (2003) A comparison of parsing technologies for the biomedical domain. *Natural Language Engineering*, 1, 1-38.
- GUEHENNEUX, F., DURET, L., CALLANAN, M. B., BOUHAS, R., HAYETTE, S., BERTHET, C., SAMARUT, C., RIMOKH, R., BIROT, A. M., WANG, Q., MAGAUD, J. P. & ROUAULT, J. P. (1997) Cloning of the mouse BTG3 gene and definition of a new gene family (the BTG family) involved in the negative control of the cell cycle. *Leukemia*, 11, 370-5.
- GUELZIM, N., BOTTANI, S., BOURGINE, P. & KEPES, F. (2002) Topological and causal structure of the yeast transcriptional regulatory network. *Nature Genetics*, 31, 60-3.
- GUO, A., HE, K., LIU, D., BAI, S., GU, X., WEI, L. & LUO, J. (2005) DATF: a database of Arabidopsis transcription factors. *Bioinformatics*, 21, 2568-9.
- GUTIERREZ-RIOS, R. M., ROSENBLUETH, D. A., LOZA, J. A., HUERTA, A. M., GLASNER, J. D., BLATTNER, F. R. & COLLADO-VIDES, J. (2003) Regulatory network of Escherichia coli: consistency between literature knowledge and microarray profiles. *Genome Research*, 13, 2435-43.
- HADSELL, D. L., TORRES, D., GEORGE, J., CAPUCO, A. V., ELLIS, S. E. & FIOROTTO, M. L. (2006) Changes in secretory cell turnover, and mitochondrial oxidative damage in the mouse mammary gland during a single prolonged lactation cycle suggest the possibility of accelerated cellular aging. *Experimental Gerontology*, 41, 271-81.
- HAKENBERG, J., PLAKE, C., ROYER, L., STROBELT, H., LESER, U. & SCHROEDER, M. (2008) Gene mention normalization and interaction extraction with context models and sentence motifs. *Genome Biology*, 9 Suppl 2, S14.
- HAN, J. & KAMBER, M. (2006) *Data Mining: Concepts and Techniques.*, Morgan Kaufmann.
- HAN, D. S., KIM, H. S., JANG, W. H., LEE, S. D. & SUH, J. K. (2004) PreSPI: design and implementation of protein-protein interaction prediction service system. *Genome Inform Ser*, 15, 171-80.
- HANISCH, D., FLUCK, J., MEVISSSEN, H. T. & ZIMMER, R. (2003) Playing biology's name game: identifying protein names in scientific text. *Proceedings of the Pacific Symposium on Biocomputing*.
- HANISCH, D., FUNDEL, K., MEVISSSEN, H. T., ZIMMER, R. & FLUCK, J. (2005) ProMiner: rule-based protein and gene entity recognition. *BMC Bioinformatics*, 6 Suppl 1, S14.

- HARRINGTON, C. A., ROSENOW, C. & RETIEF, J. (2000) Monitoring gene expression using DNA microarrays. *Current Opinions in Microbiology*, 3, 285-91.
- HART, I. C. (1972) Level of prolactin in the blood of the goat during milking, throughout lactation and over a 24-h period. *Journal of Endocrinology*, 55, 28.
- HARTMANN, P. E. (1973) Changes in the composition and yield of the mammary secretion of cows during the initiation of lactation. *Journal of Endocrinology*, 59, 231-47.
- HAO, Y., ZHU, X., HUANG, M. & LI, M. (2005) Discovering patterns to extract protein-protein interactions from the literature: Part II. *Bioinformatics*, 21, 3294-300.
- HASLAM, S. Z. & LEVELY, M. L. (1985) Estrogen responsiveness of normal mouse mammary cells in primary cell culture: association of mammary fibroblasts with estrogenic regulation of progesterone receptors. *Endocrinology*, 116, 1835-44.
- HASTIE, T., TIBSHIRANI, R., EISEN, M. B., ALIZADEH, A., LEVY, R., STAUDT, L., CHAN, W. C., BOTSTEIN, D. & BROWN, P. (2000) 'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 1, research0003.1-0003.21.
- HATZIVASSILOGLOU, V., DUBOUE, P. A. & RZHETSKY, A. (2001) Disambiguating proteins, genes, and RNA in text: a machine learning approach. *Bioinformatics*, 17 Suppl 1, S97-106.
- HAWSE, J. R., HEJTMANCIK, J. F., HUANG, Q., SHEETS, N. L., HOSACK, D. A., LEMPICKI, R. A., HORWITZ, J. & KANTOROW, M. (2003) Identification and functional clustering of global gene expression differences between human age-related cataract and clear lenses. *Molecular Vision*, 9, 515-37.
- HE, M., WANG, Y. & LI, W. (2009) PPI finder: a mining tool for human protein-protein interactions. *PLoS ONE*, 4, e4554.
- HEINRICH, K. E., BERRY, M. W. & HOMAYOUNI, R. (2008) Gene tree labeling using nonnegative matrix factorization on biomedical literature. *Computational Intelligence and Neuroscience*, 2008, Article ID: 276535.
- HENDRY, K. A., SIMPSON, K. J., NICHOLAS, K. R. & WILDE, C. J. (1998) Autocrine inhibition of milk secretion in the lactating tammar wallaby (*Macropus eugenii*). *Journal of Molecular Endocrinology*, 21, 169-77.
- HENNIGHAUSEN, L. & ROBINSON, G. W. (2001) Signaling pathways in mammary gland development. *Dev Cell*, 1, 467-75.

- HENNIGHAUSEN, L., ROBINSON, G. W., WAGNER, K. U. & LIU, W. (1997) Prolactin signaling in mammary gland development. *Journal of Biological Chemistry*, 272, 7567-9.
- HERRERO, J., DIAZ-URIARTE, R. & DOPAZO, J. (2003) Gene expression data preprocessing. *Bioinformatics*, 19, 655-6.
- HERSH, W., BHUPATIRAJU, R. T. & CORLEY, S. (2004) Enhancing access to the Bibliome: the TREC Genomics Track. *Medinfo*, 11, 773-7.
- HERSH, W., PRICE, S. & DONOHOE, L. (2000) Assessing thesaurus-based query expansion using the UMLS Metathesaurus. *Proceedings of the AMIA Annual Symposium*, 344-8.
- HERSH, W. (2005) Evaluation of biomedical text-mining systems: lessons learned from information retrieval. *Briefings in Bioinformatics*, 6, 344-56.
- HERZEL, H., BEULE, D., KIELBASA, S., KORBEL, J., SERS, C., MALIK, A., EICKHOFF, H., LEHRACH, H. & SCHUCHHARDT, J. (2001) Extracting information about cDNA arrays. *Chao*, 11, 98-107.
- HIRSCHMAN, L. (1998) The evolution of evaluation: lessons from the Message Understanding Conferences. *Information Processing and Management*, 37, 383-402.
- HIRSCHMAN, L., PARK, J. C., TSUJII, J., WONG, L. & WU, C. H. (2002) Accomplishments and challenges in literature data mining for biology. *Bioinformatics*, 18, 1553-1561.
- HOFFMANN, R., KRALLINGER, M., ANDRES, E., TAMAMES, J., BLASCHKE, C. & VALENCIA, A. (2005) Text mining for metabolic pathways, signaling cascades, and protein networks. *Sci STKE*, 2005, pe21.
- HOFFMANN, R. & VALENCIA, A. (2004) A gene network for navigating the literature. *Nature Genetics*, 36, 664.
- HOLLIDAY, G. L., BARTLETT, G. J., ALMONACID, D. E., O'BOYLE, N. M., MURRAY-RUST, P., THORNTON, J. M. & MITCHELL, J. B. (2005) MACiE: a database of enzyme reaction mechanisms. *Bioinformatics*, 21, 4315-6.
- HOLM, J., HANSEN, S. I., HOIER-MADSEN, M., KORSBAEK, L., BECKMANN, H. & JOSEFSEN, K. (2000) Ligand binding characteristics of a glycosylphosphatidyl inositol membrane-anchored HeLa cell folate receptor epitope-related to human milk folate binding protein. *Bioscience Reports*, 20, 109-18.
- HOLTER, N. S., MARITAN, A., CIEPLAK, M., FEDOROFF, N. V. & BANAVAR, J.

- R. (2001) Dynamic modeling of gene expression data. *Proceedings of the National Academy of Science U S A*, 98, 1693-1698.
- HOLTER, N. S., MITRA, M., MARITAN, A., CIEPLAK, M., BANAVAR, J. R. & FEDOROFF, N. V. (2000) Fundamental patterns underlying gene expression profiles: Simplicity from complexity. *Proceedings of the National Academy of Science U S A*, 97, 8409–8414.
- HOON, M. J. L. D., IMOTO, S. & MIYANO, S. (2002) Statistical analysis of a small set of time-ordered gene expression data using linear splines. *Bioinformatics*, 18, 1477-1485.
- HORN, F., LAU, A. L. & COHEN, F. E. (2004) Automated extraction of mutation data from the literature: application of MuteXt to G protein-coupled receptors and nuclear hormone receptors. *Bioinformatics*, 20, 557-68.
- HOU, W. J. & CHEN, H. H. (2004) Enhancing performance of protein and gene name recognizers with filtering and integration strategies. *Journal of Biomedical Informatics*, 37, 448-60.
- HRISTOVSKI, D., PETERLIN, B., MITCHELL, J. A. & HUMPHREY, S. M. (2005) Using literature-based discovery to identify disease candidate genes. *International Journal of Medical Informatics*, 74, 289-98.
- HSU, C. N., LAI, J. M., LIU, C. H., TSENG, H. H., LIN, C. Y., LIN, K. T., YEH, H. H., SUNG, T. Y., HSU, W. L., SU, L. J., LEE, S. A., CHEN, C. H., LEE, G. C., LEE, D. T., SHIUE, Y. L., YEH, C. W., CHANG, C. H., KAO, C. Y. & HUANG, C. Y. (2007) Detection of the inferred interaction network in hepatocellular carcinoma from EHCO (Encyclopedia of Hepatocellular Carcinoma genes Online). *BMC Bioinformatics*, 8, 66.
- HU, Z., MELLOR, J., WU, J., YAMADA, T., HOLLOWAY, D. & DELISI, C. (2005a) VisANT: data-integrating visual framework for biological networks and modules. *Nucleic Acids Research*, 33, W352-7.
- HU, Z. Z., NARAYANASWAMY, M., RAVIKUMAR, K. E., VIJAY-SHANKER, K. & WU, C. H. (2005b) Literature mining and database annotation of protein phosphorylation using a rule-based system. *Bioinformatics*, 21, 2759-65.
- HU, M. & QIN, Z. S. (2009) Query large scale microarray compendium datasets using a model-based bayesian approach with variable selection. *PLoS ONE*, 4, e4495.
- HUANG, M., ZHU, X., HAO, Y., PAYAN, D. G., QU, K. & LI, M. (2004) Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics*, 20, 3604-12.
- HUANG, M., DING, S., WANG, H. & ZHU, X. (2008) Mining physical protein-protein

interactions from the literature. *Genome Biology*, 9 Suppl 2, S12.

HUGHES, T. R., MARTON, M. J., JONES, A. R., ROBERTS, C. J., STOUGHTON, R., ARMOUR, C. D., BENNETT, H. A., COFFEY, E., DAI, H., HE, Y. D., KIDD, M. J., KING, A. M., MEYER, M. R., SLADE, D., LUM, P. Y., STEPANIANTS, S. B., SHOEMAKER, D. D., GACHOTTE, D., CHAKRABURTTY, K., SIMON, J., BARD, M. & FRIEND, S. H. (2000) Functional discovery via a compendium of expression profiles. *Cell*, 102, 109-26.

HUNTER, L. & COHEN, K. B. (2006) Biomedical language processing: what's beyond PubMed? *Molecular Cell*, 21, 589-94.

HURLEY, W. L., BLATCHFORD, D. R., HENDRY, K. A. & WILDE, C. J. (1994) Extracellular matrix and mouse mammary cell function: comparison of substrata in culture. *In Vitro Cellular & Developmental Biology. Animal.*, 30A, 529-38.

HUSEBY, N. E., ASARE, N., WETTING, S., MIKKELSEN, I. M., MORTENSEN, B., SVEINBJORNSSON, B. & WELLMAN, M. (2003) Nitric oxide exposure of CC531 rat colon carcinoma cells induces gamma-glutamyltransferase which may counteract glutathione depletion and cell death. *Free Radical Research*, 37, 99-107.

HUSMEIER, D. (2003) Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19, 2271-82.

IGARASHI, H., KUWATA, N., KIYOTA, K., SUMITA, K., SUDA, T., ONO, S., BAUER, S. R. & SAKAGUCHI, N. (2001) Localization of recombination activating gene 1/green fluorescent protein (RAG1/GFP) expression in secondary lymphoid organs after immunization with T-dependent antigens in rag1/gfp knockin mice. *Blood*, 97, 2680-7.

ILKBAHAR, Y. N., THORDARSON, G., CAMARILLO, I. G. & TALAMANTES, F. (1999) Differential expression of the growth hormone receptor and growth hormone-binding protein in epithelia and stroma of the mouse mammary gland at various physiological stages. *Journal of Endocrinology*, 161, 77-87.

IRIZARRY, R. A., BOLSTAD, B. M., COLLIN, F., COPE, L. M., HOBBS, B. & SPEED, T. P. (2003) Summaries of Affymetrix GeneChip probe level data. *Nucleic Acids Research*, 31, e15.

ISHII, T., ISHIMORI, H., MORI, K., UTO, T., FUKUDA, K., URASHIMA, T. & NISHIMURA, M. (2007) Bovine lactoferrin stimulates anchorage-independent cell growth via membrane-associated chondroitin sulfate and heparan sulfate proteoglycans in PC12 cells. *Journal of Pharmacological Science*, 104, 366-73.

JACOBSON, E. A., JAMES, K. A., NEWMARK, H. L. & CARROLL, K. K. (1989)

Effects of dietary fat, calcium, and vitamin D on growth and mammary tumorigenesis induced by 7,12-dimethylbenz(a)anthracene in female Sprague-Dawley rats. *Cancer Research*, 49, 6300-3.

JACQUEMN, C. (2001) *Spotting and discovering terms through natural language processing*, Cambridge, MA, MIT Press.

JAIS, A. M., MCCULLOCH, R. & CROFT, K. (1994) Fatty acid and amino acid composition in haruan as a potential role in wound healing. *General Pharmacology*, 25, 947-50.

JANG, H., LIM, J., LIM, J. H., PARK, S. J., LEE, K. C. & PARK, S. H. (2006) Finding the evidence for protein-protein interactions from PubMed abstracts. *Bioinformatics*, 22, e220-6.

JEFFERY, I. B., MADDEN, S. F., MCGETTIGAN, P. A., PERRIERE, G., CULHANE, A. C. & HIGGINS, D. G. (2007) Integrating transcription factor binding site information with gene expression datasets. *Bioinformatics*, 23, 298-305.

JELIER, R., JENSTER, G., DORSSERS, L. C., VAN DER EIJK, C. C., VAN MULLIGEN, E. M., MONS, B. & KORS, J. A. (2005) Co-occurrence based meta-analysis of scientific texts: retrieving biological relationships between genes. *Bioinformatics*, 21, 2049-58.

JELIER, R., SCHUEMIE, M. J., VELDHoven, A., DORSSERS, L. C., JENSTER, G. & KORS, J. A. (2008) Anni 2.0: a multipurpose text-mining tool for the life sciences. *Genome Biology*, 9, R96.

JENSEN, L. J., SARIC, J. & BORK, P. (2006) Literature mining for the biologist: from information retrieval to biological discovery. *Nature Review Genetics*, 7, 119-29.

JENSSEN, T. K., LAEGREID, A., KOMOROWSKI, J. & HOVIG, E. (2001) A literature network of human genes for high-throughput analysis of gene expression. *Nature Genetics*, 28, 21-8.

JERUSS, J. S., SANTIAGO, J. Y. & WOODRUFF, T. K. (2003) Localization of activin and inhibin subunits, receptors and SMADs in the mouse mammary gland. *Molecular and Cellular Endocrinology*, 203, 185-96.

JIA, Z., MOULSON, C. L., PEI, Z., MINER, J. H. & WATKINS, P. A. (2007) Fatty Acid Transport Protein 4 Is the Principal Very Long Chain Fatty Acyl-CoA Synthetase in Skin Fibroblasts. *Journal of Biological Chemistry*, 282, 20573-20583.

JIMENO, A., JIMENEZ-RUIZ, E., LEE, V., GAUDAN, S., BERLANGA, R. & REBHOLZ-SCHUHMANN, D. (2008) Assessment of disease named entity recognition on a corpus of annotated sentences. *BMC Bioinformatics*, 9 Suppl 3, S3.



- JIN, B., MULLER, B., ZHAI, C. & LU, X. (2008) Multi-label literature classification based on the Gene Ontology graph. *BMC Bioinformatics*, 9, 525.
- JIRIKOWSKI, G. F. (1992) Oxytocinergic neuronal systems during mating, pregnancy, parturition, and lactation. *Annals of the New York Academy of Science*, 652, 253-70.
- JUAN, H. F. & HUANG, H. C. (2007) Bioinformatics: microarray data clustering and functional classification. *Methods in Molecular Biology*, 382, 405-16.
- JUERGENS, W. G., STOCKDALE, F. E., TOPPER, Y. J. & ELIAS, J. J. (1965) Hormone-dependent differentiation of mammary gland in vitro. *Proceedings of the National Academy of Science U S A*, 54, 629-34.
- KAAS, Q., WESTERMANN, J. C., HALAI, R., WANG, C. K. & CRAIK, D. J. (2008) ConoServer, a database for conopeptide sequences and structures. *Bioinformatics*, 24, 445-6.
- KALISH, J. A., WILLIS, D. J., LI, C., LINK, J. J., DEUTSCH, E. R., CONTRERAS, M. A., QUIST, W. C. & LOGERFO, F. W. (2004) Temporal genomics of vein bypass grafting through oligonucleotide microarray analysis. *Journal of Vascular Surgery*, 39, 645-54.
- KANO, Y., NGUYEN, N., SAETRE, R., YOSHIDA, K., MIYAO, Y., TSURUOKA, Y., MATSUBAYASHI, Y., ANANIADOU, S. & TSUJII, J. (2008) Filling the gaps between tools and users: a tool comparator, using protein-protein interaction as an example. *Pacific Symposium on Biocomputing*, 616-27.
- KAROPKA, T., SCHEEL, T., BANSEMER, S. & GLASS, A. (2004) Automatic construction of gene relation networks using text mining and gene expression data. *Medical Informatics and the Internet in Medicine*, 29, 169-83.
- KASTURI, J., ACHARYA, R. & RAMANATHAN, M. (2003) An information theoretic approach for analyzing temporal patterns of gene expression. *Bioinformatics*, 19, 449-458.
- KEOUGH, E. M. & WOOD, B. G. (1979) Mammary gland development during pregnancy in the dwarf mouse mutant, little. *Tissue Cell*, 11, 773-80.
- KERR, M. K. & CHURCHILL, G. A. (2001) Statistical design and the analysis of gene expression microarrays. *Genetical Research*, 77, 123-128.
- KEYS, J. E., VAN ZYL, J. P. & FARRELL, H. M., JR. (1994) In vitro DNA synthesis as indicator of mammary epithelial cell division: [14C]thymidine uptake versus flow cytometry cell cycle analysis. *In Vitro Cell Dev Biol Anim*, 30A, 50-5.
- KEYS, J. E., CIFRIAN, E., GUIDRY, A. J. & FARRELL, H. M. (1997) Bovine

- mammary explant versus primary cell cultures: effect of bovine somatotropin and insulin-like growth factor-I on DNA content and protein synthesis. *In Vitro Cellular and Developmental Biology of Animals*, 33, 206-11.
- KIM, J. D., OHTA, T. & TSUJII, J. (2008) Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9, 10.
- KIM, J., SEO, J., LEE, Y. S. & KIM, S. (2005) TFExplorer: integrated analysis database for predicted transcription regulatory elements. *Bioinformatics*, 21, 548-50.
- KIM, S. Y., IMOTO, S. & MIYANO, S. (2003) Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in Bioinformatics*, 4, 228-35.
- KIM, J. D., OHTA, T., TATEISI, Y. & TSUJII, J. (2003) GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19, i180-i182.
- KIMURA, S., SONODA, K., YAMANE, S., MAEDA, H., MATSUMURA, K. & HATAKEYAMA, M. (2008) Function approximation approach to the inference of reduced NGnet models of genetic networks. *BMC Bioinformatics*, 9, 23.
- KLEINBERG, D. L. (1997) Early mammary development: growth hormone and IGF-1. *Journal of Mammary Gland Biology and Neoplasia*, 2, 49-57.
- KODRATOFF, Y., DIMULESCU, A. & AMRANI, A. (2005) Man-Machine Cooperation in Retrieving Knowledge from Technical Texts. *AAAI 2005 Symposium on Mixed-Initiative Problem-Solving Assistants*.
- KOROTKIY, M., MIDDELBURG, R., DEKKER, H., VAN HARMELEN, F. & LANKELMA, J. (2004) A tool for gene expression based PubMed search through combining data sources. *Bioinformatics*, 20, 1980-2.
- KOU, Z., COHEN, W. W. & MURPHY, R. F. (2005) High-recall protein entity recognition using a dictionary. *Bioinformatics*, 21 Suppl 1, i266-73.
- KRALLINGER, M., LEITNER, F., RODRIGUEZ-PENAGOS, C. & VALENCIA, A. (2008a) Overview of the protein-protein interaction annotation extraction task of BioCreative II. *Genome Biology*, 9 Suppl 2, S4.
- KRALLINGER, M., VALENCIA, A. & HIRSCHMAN, L. (2008b) Linking genes to literature: text mining, information extraction, and retrieval applications for biology. *Genome Biology*, 9 Suppl 2, S8.
- KRATOCHWIL, K. & SCHWARTZ, P. (1976) Tissue interaction in androgen response of embryonic mammary rudiment of mouse: identification of target tissue for testosterone. *Proceedings of the National Academy of Science U S A*, 73, 4041-4.

- KRAUTHAMMER, M. & NENADIC, G. (2004) Term identification in the biomedical literature. *Journal of Medical Bioinformatics*, 37, 512-26.
- KROLL, T. C. & WOLFL, S. (2002) Ranking: a closer look on globalisation methods for normalization of gene expression arrays. *Nucleic Acid Research*, 30, e50.
- KUDO, T. & MATSUMOTO, Y. (2000) Use of support vector learning for chunk identification. *4th Conference on CoNLL-2000 and LLL-2000*.
- KUHN, N. J. (1969) Progesterone withdrawal as the lactogenic trigger in the rat. *Journal of Endocrinology*, 44, 39-54.
- KULSKI, J. K., NICHOLAS, K. R., TOPPER, Y. J. & QASBA, P. (1983a) Essentiality of insulin and prolactin for accumulation of rat casein mRNAs. *Biochemical and Biophysical Research Communication*, 116, 994-999.
- KULSKI, J. K., TOPPER, Y. J., CHOMCZYNSKI, P. & QASBA, P. (1983b) An essential role for glucocorticoid in casein gene expression in rat mammary explants. *Biochemical and Biophysical Research Communication*, 114, 380-387.
- KUMAR, R., VADLAMUDI, R. K. & ADAM, L. (2000) Apoptosis in mammary gland and cancer. *Endocrine-Related Cancer*, 7, 257-69.
- KUPIEC, J. (1992) Robust part-of-speech tagging using a Hidden Markov Model. *Computer Speech and Language*, 6.
- KUSAKARI, Y., OGAWA, E., OWADA, Y., KITANAKA, N., WATANABE, H., KIMURA, M., TAGAMI, H., KONDO, H., AIBA, S. & OKUYAMA, R. (2006) Decreased keratinocyte motility in skin wound on mice lacking the epidermal fatty acid binding protein gene. *Molecular and Cellular Biochemistry*, 284, 183-8.
- KWON, H. C., KIM, S. H., ROH, M. S., KIM, J. S., LEE, H. S., CHOI, H. J., JEONG, J. S., KIM, H. J. & HWANG, T. H. (2004) Gene expression profiling in lymph node-positive and lymph node-negative colorectal cancer. *Diseases of the Colon & Rectum*, 47, 141-52.
- KYRIAKOU, S. Y. & KUHN, N. J. (1973) Lactogenesis in the diabetic rat. *Journal of Endocrinology*, 59, 199-200.
- LAI, Y. (2008) Genome-wide co-expression based prediction of differential expressions. *Bioinformatics*, 24, 666-73.
- LAKSHMANAN, L. V. S., SADRI, F. & SUBRAMANIAN, S. N. (2001) SchemaSQL - An extension to SQL for multidatabase interoperability. *ACM Transactions on Database Systems*, 26, 476-519.
- LEE, L. C., HORN, F. & COHEN, F. E. (2007) Automatic Extraction of Protein Point

- Mutations Using a Graph Bigram Association. *PLoS Computational Biology*, 3, e16.
- LEE, H. K., HSU, A. K., SAJDAK, J., QIN, J. & PAVLIDIS, P. (2004) Coexpression analysis of human genes across many microarray data sets. *Genome Research*, 14, 1085-94.
- LEE, D. K., PARK, J. W., KIM, Y. J., KIM, J., LEE, Y. & KIM, J. S. (2003) Toward a functional annotation of the human genome using artificial transcription factors. *Genome Research*, 13, 2708-16.
- LEE, H., YI, G. S. & PARK, J. C. (2008) E3Miner: a text mining tool for ubiquitin-protein ligases. *Nucleic Acids Research*, 36, W416-22.
- LEEK, T. R. (1997) Information extraction using Hidden Markov Model. *Department of Computer Science*. San Diego, University of California.
- LEMAY, D. G., NEVILLE, M. C., RUDOLPH, M. C., POLLARD, K. S. & GERMAN, J. B. (2007) Gene regulatory networks in lactation: identification of global principles using bioinformatics. *BMC Systems Biology*, 1, 56.
- LEROY, G. & CHEN, H. (2002) Filling preposition-based templates to capture information from medical abstracts. *Pacific Symposium on Biocomputing*, 7, 350-361.
- LESER, U. & HAKENBERG, J. (2005) What makes a gene name? Named entity recognition in the biomedical literature. *Briefings on Bioinformatics*, 6, 357-69.
- LI, H. & GUI, J. (2006) Gradient directed regularization for sparse Gaussian concentration graphs, with applications to inference of genetic networks. *Biostatistics*, 7, 302-17.
- LI, H. & HONG, F. (2001) Cluster-Rasch models for microarray gene expression data. *Genome Biology*, 2, research0031.1-0031.13.
- LI, B. & GALLIN, W. J. (2004) VKCDB: voltage-gated potassium channel database. *BMC Bioinformatics*, 5, 3.
- LI, Q. & WU, Y. F. (2006) Identifying important concepts from medical documents. *Journal of Biomedical Informatics*, 39, 668-79.
- LI, X., CHEN, H., HUANG, Z., SU, H. & MARTINEZ, J. D. (2007) Global mapping of gene/protein interactions in PubMed abstracts: A framework and an experiment with P53 interactions. *Journal of Biomedical Informatics*.
- LI, J. L., LI, M. X., DENG, H. Y., DUFFY, P. E. & DENG, H. W. (2005) PhD: a web database application for phenotype data management. *Bioinformatics*, 21, 3443-

- LI, H., LIU, Y., SHIN, S., SUN, Y., LORING, J. F., MATTSON, M. P., RAO, M. S. & ZHAN, M. (2006a) Transcriptome coexpression map of human embryonic stem cells. *BMC Genomics*, 7, 103.
- LI, S. & ROSEN, J. M. (1995) Nuclear factor I and mammary gland factor (STAT5) play a critical role in regulating rat whey acidic protein gene expression in transgenic mice. *Molecular Cell Biology*, 15, 2063-70.
- LI, S., WU, L. & ZHANG, Z. (2006b) Constructing biological networks through combined literature mining and microarray analysis: a LMMA approach. *Bioinformatics*, 22, 2143-50.
- LIAO, M. J., ZHANG, C. C., ZHOU, B., ZIMONJIC, D. B., MANI, S. A., KABA, M., GIFFORD, A., REINHARDT, F., POPESCU, N. C., GUO, W., EATON, E. N., LODISH, H. F. & WEINBERG, R. A. (2007) Enrichment of a population of mammary gland cells that form mammospheres and have in vivo repopulating activity. *Cancer Research*, 67, 8131-8.
- LIN, C. Y., CHEN, C. L., CHO, C. S., WANG, L. M., CHANG, C. M., CHEN, P. Y., LO, C. Z. & HSIUNG, C. A. (2005) hp-DPI: Helicobacter pylori database of protein interactomes--embracing experimental and inferred interactions. *Bioinformatics*, 21, 1288-90.
- LING, M. H. (2006) An Anthological Review of Research Utilizing MontyLingua, a Python-Based End-to-End Text Processor. *The Python Papers*, 1, 5-12.
- LING, M. H., LEFEVRE, C., NICHOLAS, K. R. & LIN, F. (2007) Re-construction of Protein-Protein Interaction Pathways by Mining Subject-Verb-Objects Intermediates. *Second IAPR Workshop on Pattern Recognition in Bioinformatics (PRIB 2007)*. Singapore, Springer-Verlag.
- LIU, H. (2004) MontyLingua: An end-to-end natural language processor with common sense.
- LIU, H. & FRIEDMAN, C. (2003) Mining terminological knowledge in large biomedical corpora. *Proceedings of the Pacific Symposium on Biocomputing*, 8, 415-426.
- LIU, H., LI, X., YOON, V. & CLARKE, R. (2008) Annotating breast cancer microarray samples using ontologies. *AMIA Annual Symposium Proceedings*, 414-8.
- LIU, H. & SINGH, P. (2004a) Commonsense reasoning in and over natural language. *8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES'2004)*. Wellington, New Zealand, Springer.

- LIU, H. & SINGH, P. (2004b) ConceptNet: A Practical Commonsense Reasoning Toolkit. *BT Technology Journal*, 22, 211-226.
- LIU, H., HU, Z. Z., TORII, M., WU, C. & FRIEDMAN, C. (2006a) Quantitative assessment of dictionary-based protein named entity tagging. *Journal of the American Medical Informatics Association*, 13, 497-507.
- LIU, Y., LI, J., SAM, L., GOH, C. S., GERSTEIN, M. & LUSSIER, Y. A. (2006b) An integrative genomic approach to uncover molecular mechanisms of prokaryotic traits. *PLoS Computational Biology*, 2, e159.
- LIU, X., ROBINSON, G. W., WAGNER, K. U., GARETT, L., WYNshaw-BORIS, A. & HENNIGHAUSEN, L. (1997) Stat5a is mandatory for adult mammary gland development and lactogenesis. *Genes and Development*, 11, 179-186.
- LIU, Y. F. & YANG, U. C. (2004) SCA db: spinocerebellar ataxia candidate gene database. *Bioinformatics*, 20, 2656-61.
- LKHIDER, M., PETRIDOU, B., AUBOURG, A. & OLLIVIER-BOUSQUET, M. (2001) Prolactin signalling to milk protein secretion but not to gene expression depends on the integrity of the Golgi region. *Journal of Cell Science*, 114, 1883-91.
- LOMIZE, M. A., LOMIZE, A. L., POGOZHEVA, I. D. & MOSBERG, H. I. (2006) OPM: orientations of proteins in membranes database. *Bioinformatics*, 22, 623-5.
- LU, L. F., GAVIN, M. A., RASMUSSEN, J. P. & RUDENSKY, A. Y. (2007) G protein-coupled receptor 83 is dispensable for the development and function of regulatory T cells. *Molecular Cell Biology*, 27, 8065-72.
- LUSSIER, Y., BORLAWSKY, T., RAPPAPORT, D., LIU, Y. & FRIEDMAN, C. (2006) PhenoGO: assigning phenotypic context to gene ontology annotations with natural language processing. *Proceedings of the Pacific Symposium of Biocomputing*, 64-75.
- LYNN, K. S., HSU, W. L., LI, L. L., LIN, Y. J., WANG, C. H., SHENG, S. H., LIN, J. H., LIAO, W. & PAN, W. H. (2009) A Neural Network Model for Constructing Endophenotypes of Common Complex Diseases: -An Application to Male Young-onset Hypertension Microarray Data. *Bioinformatics*.
- MA'AYAN, A., GARDINER, K. & IYENGAR, R. (2006) The cognitive phenotype of Down syndrome: insights from intracellular network analysis. *NeuroRx*, 3, 396-406.
- MACKIE, T. R., DWYER, D. A., INGVARTSEN, K. L., CHOUINARD, P. Y., ROSS, D. A. & BAUMAN, D. E. (2000) Effects of insulin and postruminal supply of protein on use of amino acids by mammary gland for milk protein synthesis.

- MAERE, S., HEYMANS, K. & KUIPER, M. (2005) BiNGO: a Cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks. *Bioinformatics*, 21, 3448-9.
- MAGKRIOTI, C. K., SPYROPOULOS, I. C., ICONOMIDOU, V. A., WILLIS, J. H. & HAMODRAKAS, S. J. (2004) cuticleDB: a relational database of Arthropod cuticular proteins. *BMC Bioinformatics*, 5, 138.
- MAGUITMAN, A. G., RECHTSTEINER, A., VERSPOOR, K., STRAUSS, C. E. & ROCHA, L. M. (2006) Large-scale testing of bibliome informatics using Pfam protein families. *Proceedings of the Pacific Symposium of Biocomputing*, 76-87.
- MAJOROS, W., SUBRAMANIAN, G. & YANDELL, M. (2003) Identification of key concepts in biomedical literature using a modified Markov heuristic. *Bioinformatics*, 19, 402-407.
- MAJUMDER, G. C. & TURKINGTON, R. W. (1971) Hormonal regulation of protein kinases and adenosine 3', 5'-monophosphate-binding protein in developing mammary gland. *Journal of Biological Chemistry*, 246, 5545-5554.
- MALEWSKI, T., GAJEWSKA, M., ZEBROWSKA, T. & ZWIERZCHOWSKI, L. (2002) Differential induction of transcription factors and expression of milk protein genes by prolactin and growth hormone in the mammary gland of rabbits. *Growth Hormone & IGF Research*, 12, 41-53.
- MALIK, R., FRANKE, L. & SIEBES, A. (2006) Combination of text-mining algorithms increases the performance. *Bioinformatics*.
- MANDUCHI, E., GRANT, G. R., MCKENZIE, S. E., OVERTON, G. C., SURREY, S. & STOECKERT, C. J. (2000) Generation of patterns from gene expression data by assigning confidence to differentially expressed genes. *Bioinformatics*, 16, 685-198.
- MAO, C., QIU, J., WANG, C., CHARLES, T. C. & SOBRAL, B. W. (2005) NodMutDB: a database for genes and mutants involved in symbiosis. *Bioinformatics*, 21, 2927-9.
- MARAZIOTIS, I. A., DRAGOMIR, A. & BEZERIANOS, A. (2007) Gene networks reconstruction and time-series prediction from microarray data using recurrent neural fuzzy networks. *IET System Biology*, 1, 41-50.
- MARCUS, M. P., SANTORINI, B. & MARCINKIEWICZ, M. A. (1993) Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19, 313-330.

- MARGOLIN, A. A., WANG, K., LIM, W. K., KUSTAGI, M., NEMENMAN, I. & CALIFANO, A. (2006) Reverse engineering cellular networks. *Nature Protocol*, 1, 662-71.
- MARGOLIS, D. J., BILKER, W., BOSTON, R., LOCALIO, R. & BERLIN, J. A. (2002) Statistical characteristics of area under the receiver operating characteristic curve for a simple prognostic model using traditional and bootstrapped approaches. *Journal of Clinical Epidemiology*, 55, 518-24.
- MARKUS, R., SRINIVAS, V., SAMUEL, A., JOHAN, S. & JARI, H. K. (2004) ACID: a database for microarray clone information. *Bioinformatics*, 20, 2305.
- MARTINEZ-BUENO, M., MOLINA-HENARES, A. J., PAREJA, E., RAMOS, J. L. & TOBES, R. (2004) BacTregulators: a database of transcriptional regulators in bacteria and archaea. *Bioinformatics*, 20, 2787-91.
- MASSEROLI, M., KILICOGU, H., LANG, F. M. & RINDFLESCH, T. C. (2006) Argument-predicate distance as a filter for enhancing precision in extracting predications on the genetic etiology of disease. *BMC Bioinformatics*, 7, 291.
- MASTER, S. R., HARTMAN, J. L., D'CRUZ, C. M., MOODY, S. E., KEIPER, E. A., HA, S. I., COX, J. D., BELKA, G. K. & CHODOSH, L. A. (2002) Functional microarray analysis of mammary organogenesis reveals a developmental role in adaptive thermogenesis. *Molecular Endocrinology*, 16, 1185-1203.
- MATSUZAWA, H. & FUKUDA, T. (2000) Mining structured association patterns from database. *4th Pacific and Asia International Conference on Knowledge Discovery and Data Mining (PAKDD-2000)*.
- MATUSIK, R. J. & ROSEN, J. M. (1978) Prolactin induction of casein mRNA in organ culture. A model system for studying peptide hormone regulation of gene expression. *Journal of Biological Chemistry*, 253, 2343-7.
- MCDONALD, R. & PEREIRA, F. (2005) Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics*, 6 Suppl 1, S6.
- MCMANAMAN, J. L. & NEVILLE, M. C. (2003) Mammary physiology and milk secretion. *Advanced Drug Delivery Review*, 55, 629-41.
- MENZIES, K. K., LEFEVRE, C., MACMILLAN, K. L. & NICHOLAS, K. R. (2009) Insulin regulates milk protein synthesis at multiple levels in the bovine mammary gland. *Functional & Integrative Genomics*, 9, 197-217.
- MIKA, S. & ROST, B. (2004) NLProt: extracting protein names and sequences from papers. *Nucleic Acids Research*, 32, W634-7.
- MIKHEEV, A. (1997) Automatic rule induction for unknown word guessing.



- MILLS, E. S. & TOPPER, Y. J. (1969) Mammary alveolar epithelial cells: effect of hydrocortisone on ultrastructure. *Science*, 165, 1127-8.
- MILLS, E. S. & TOPPER, Y. J. (1970) Some ultrastructural effects of insulin, hydrocortisone, and prolactin on mammary gland explants. *Journal of Cell Biology*, 44, 310-28.
- MINNEN, G., CARROLL, J. & PEARCE, D. (2001) Applied morphological processing of English. *Natural Language Engineering*, 7, 207-223.
- MIOTTO, O., TAN, T. W. & BRUSIC, V. (2005) Supporting the curation of biological databases with reusable text mining. *Genome Informatics*, 16, 32-44.
- MIYAO, Y., SAGAE, K., SAETRE, R., MATSUZAKI, T. & TSUJII, J. (2009) Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, 25, 394-400.
- MONTANER, D., TARRAGA, J., HUERTA-CEPAS, J., BURGUET, J., VAQUERIZAS, J. M., CONDE, L., MINGUEZ, P., VERA, J., MUKHERJEE, S., VALLS, J., PUJANA, M. A., ALLOZA, E., HERRERO, J., AL-SHAHROUR, F. & DOPAZO, J. (2006) Next station in microarray data analysis: GEPAS. *Nucleic Acids Research*, 34, W486-91.
- MORAN, L. B., DUKE, D. C. & GRAEBER, M. B. (2007) The microglial gene regulatory network activated by interferon-gamma. *Journal of Neuroimmunology*, 183, 1-6.
- MORI, R., KONDO, T., NISHIE, T., OHSHIMA, T. & ASANO, M. (2004) Impairment of skin wound healing in beta-1,4-galactosyltransferase-deficient mice with reduced leukocyte recruitment. *American Journal of Pathology*, 164, 1303-14.
- MORRIS, R. T., O'CONNOR, T. R. & WYRICK, J. J. (2008) Osiris: an integrated promoter database for *Oryza sativa* L. *Bioinformatics*, 24, 2915-7.
- MULLER, H. M., KENNY, E. E. & STERNBERG, P. W. (2004) Textpresso: an ontology-based information retrieval and extraction system for biological literature. *PLoS Biology*, 2, e309.
- MULLER, M., MARKO, K., DAUMKE, P., PAETZOLD, J., ROESNER, A. & KLAR, R. (2007) Biomedical data mining in clinical routine: expanding the impact of hospital information systems. *Medinfo*, 12, 340-4.
- MURPHY, G., ARIYANAYAGAM, A. D. & KUHN, N. J. (1973) Progesterone and the metabolic control of the lactose biosynthetic pathway during lactogenesis in the rat. *Biochemical Journal*, 136, 1105-16.

- NACHMAN, I., REGEV, A. & FRIEDMAN, N. (2004) Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, 20 Suppl 1, I248-I256.
- NAGAIHAH, K., FRANKLYN F. BORLANDER, J., NICHOLAS, K. R., TAKEMOTO, T. & TOPPER, Y. J. (1981) Prolactin-induced accumulation of casein mRNA in mouse mammary explants: a selective role of glucocorticoid. *Biochemical and Biophysical Research Communication*, 98, 380-387.
- NASUKAWA, T. & NAGONO, T. (2001) Text analysis and knowledge mining system. *IBM Systems Journal*, 40, 967-984.
- NATARAJAN, J., BERRAR, D., DUBITZKY, W., HACK, C., ZHANG, Y., DESESA, C., VAN BROCKLYN, J. R. & BREMER, E. G. (2006) Text mining of full-text journal articles combined with gene expression analysis reveals a relationship between sphingosine-1-phosphate and invasiveness of a glioblastoma cell line. *BMC Bioinformatics*, 7, 373.
- NATARAJAN, J., BERRAR, D., HACK, C. J. & DUBITZKY, W. (2005) Knowledge discovery in biology and biotechnology texts: a review of techniques, evaluation strategies, and applications. *Critical Reviews in Biotechnology*, 25, 31-52.
- NATARAJAN, J. & GANAPATHY, J. (2007) Functional gene clustering via gene annotation sentences, MeSH and GO keywords from biomedical literature. *Bioinformation*, 2, 185-93.
- NATIONAL LIBRARY OF MEDICINE (2003) *UMLS Knowledge Sources*.
- NAYLOR, M. J., OAKES, S. R., GARDINER-GARDEN, M., HARRIS, J., BLAZEK, K., HO, T. W., LI, F. C., WYNICK, D., WALKER, A. M. & ORMANDY, C. J. (2005) Transcriptional changes underlying the secretory activation phase of mammary gland development. *Molecular Endocrinology*, 19, 1868-83.
- NEFF, M. S., BYRD, R. J. & BOGURAEV, B. K. (2004) The Talent system: TEXTTRACT architecture and data model. *Natural Language Engineering*, 10, 307-326.
- NEVILLE, M. C. & MORTON, J. (2001) Physiology and endocrine changes underlying human lactogenesis II. *Journal of Nutrition*, 131, 3005S-8S.
- NENADIC, G., SPASIC, I. & ANANIADOU, S. (2004) Mining biomedical abstracts: What is in a term? IN SU, K. Y. (Ed.) *Natural Language Processing - IJCNLP 2004*. Berlin, Springer.
- NEWMAN, D., HETTICH, S., BLAKE, C. & MERZ, C. (1998) UCI Repository of machine learning databases. Irvine, CA, University of California, Department of Information and Computer Science.

- NICHOLAS, K. R., COLLET, C., JOSEPH, R. & SANKARAN, L. (1991) Hormone-responsive survival of mammary gland explants from the pregnant tammar wallaby (*Macropus eugenii*) in the absence of exogenous hormones and growth factors. *Comparative Biochemistry and Physiology A: Comparative Physiology*, 100, 163-7.
- NICHOLAS, K. R. & HARTMANN, P. E. (1981a) Progesterone control of the initiation of lactose synthesis in the rat. *Australian Journal of Biological Science*, 34, 435-43.
- NICHOLAS, K. R. & HARTMANN, P. E. (1981b) The foetoplacental unit and the initiation of lactation in the rat. *Australian Journal of Biological Science*, 34, 455-61
- NICHOLAS, K. R., SANKARAN, L. & TOPPER, Y. J. (1983) A unique and essential role for insulin in the phenotypic expression of rat mammary epithelial cells unrelated to its function in cell maintenance. *Biochemica et Biophysica Acta*, 763, 309-314.
- NICHOLAS, K. R. & TOPPER, Y. J. (1983) Anti-insulin receptor serum mimics the developmental role of insulin in mouse mammary explants. *Biochemical and Biophysical Research Communication*, 111, 988-993.
- NISHIKAWA, S., MOORE, R. C., NONOMURA, N. & OKA, T. (1994) Progesterone and EGF inhibit mouse mammary gland prolactin receptor and beta-casein gene expression. *American Journal of Physiology - Cell Physiology*, 267, C1467-72.
- NOVICHKOVA, S., EGOROV, S. & DARASELIA, N. (2003) MedScan, a natural language processing engine for MEDLINE abstracts. *Bioinformatics*, 19, 1699-1706.
- NUEDA, M. J., CONESA, A., WESTERHUIS, J. A., HOEFSLOOT, H. C., SMILDE, A. K., TALON, M. & FERRER, A. (2007) Discovering gene expression patterns in time course microarray experiments by ANOVA-SCA. *Bioinformatics*, 23, 1792-800.
- O'DRISCOLL, L., MCMORROW, J., DOOLAN, P., MCKIERNAN, E., MEHTA, J. P., RYAN, E., GAMMELL, P., JOYCE, H., O'DONOVAN, N., WALSH, N. & CLYNES, M. (2006) Investigation of the molecular profile of basal cell carcinoma using whole genome microarrays. *Molecular Cancer*, 5, 74.
- OHTA, T., TATEISI, Y., MIMA, H. & TSUJII, J. (2002) The GENIA corpus: an annotated research abstract corpus in molecular biology domain. *Human Language Technology Conference*.
- OKA, T., PERRY, J. W. & TOPPER, Y. J. (1974) Changes in insulin responsiveness during development of mammary epithelium. *Journal of Cell Biology*, 62, 550-6.
- OKA, T. & TOPPER, Y. J. (1972) Is prolactin mitogenic for mammary epithelium? *Proceedings of the National Academy of Science U S A*, 69, 1693-6.

- OKAMOTO, S., YAMANISHI, Y., EHIRA, S., KAWASHIMA, S., TONOMURA, K. & KANEHISA, M. (2007) Prediction of nitrogen metabolism-related genes in *Anabaena* by kernel-based network analysis. *Proteomics*, 7, 900-909.
- OKAZAKI, N. & ANANIADOU, S. (2006) Building an abbreviation dictionary using a term recognition approach. *Bioinformatics*, 22, 3089-95.
- ONO, M. & OKA, T. (1980) alpha-Lactalbumin-casein induction in virgin mouse mammary explants: dose-dependent differential action of cortisol. *Science*, 207, 1367-9.
- OSBORNE, J. D., LIN, S., ZHU, L. & KIBBE, W. A. (2007) Mining biomedical data using MetaMap Transfer (MMtx) and the Unified Medical Language System (UMLS). *Methods in Molecular Biology*, 408, 153-69.
- OVASKA, K., LAAKSO, M. & HAUTANIEMI, S. (2008) Fast Gene Ontology based clustering for microarray experiments. *BioData Mining*, 1, 11.
- PALANISWAMY, S. K., JIN, V. X., SUN, H. & DAVULURI, R. V. (2005) OMGProm: a database of orthologous mammalian gene promoters. *Bioinformatics*, 21, 835-6.
- PANDEY, R., GURU, R. K. & MOUNT, D. W. (2004) Pathway Miner: extracting gene association networks from molecular pathways for predicting the biological significance of gene expression microarray data. *Bioinformatics*, 20, 2156-8.
- PANG, W. & HARTMANN, P. (2007) Initiation of human lactation: secretory differentiation and activation. *Journal of Mammary Gland Biology and Neoplasia*, 12, 211-21.
- PAVLIDIS, P. (2003) Using ANOVA for gene selection from microarray studies of the nervous system. *Methods*, 31, 282-9.
- PAVLIDIS, P., WESTON, J., CAI, J. & GRUNDY, W. N. (2001) Gene functional classification from heterogeneous data. *Proceedings of the Fifth International Conference on Research in Computational Molecular Biology*.
- PAWLICKI, S., LE BECHEC, A. & DELAMARCHE, C. (2008) AMYPdb: a database dedicated to amyloid precursor proteins. *BMC Bioinformatics*, 9, 273.
- PEAKER, M. & WILDE, C. J. (1996) Feedback control of milk secretion from milk. *Journal of Mammary Gland Biology and Neoplasia*, 1, 307-15.
- PESHKIN, L. & SAVOVA, V. (2003) Why Build Another Part-of-Speech Tagger? A Minimalist Approach. *Recent Advances in Natural Language Processing (RANLP-2003)*. Borovets, Bulgaria.

- PIROOZNIA, M. & DENG, Y. (2006) SVM Classifier - a comprehensive java interface for support vector machine classification of microarray data. *BMC Bioinformatics*, 7 Suppl 4, S25.
- PLAKE, C., HAKENBERG, J. & LESER, U. (2005) Optimizing syntax patterns for discovering protein-protein interactions. *ACM Symposium on Applied Computing*. ACM Press.
- PLUM, L., BELGARDT, B. F. & BRUNING, J. C. (2006) Central insulin action in energy and glucose homeostasis. *Journal of Clinical Investigation*, 116, 1761-6.
- POYET, P., HENNING, S. J. & ROSEN, J. M. (1989) Hormone-dependent beta-casein mRNA stabilization requires ongoing protein synthesis. *Molecular Endocrinology*, 3, 1961-8.
- PRANGE, J. D. (1996) Evaluation driven research: the foundation of the TIPSTER text program. *Tipster Text Program Phase II*. Vienna, Virginia, Association of Computational Linguistics.
- PRATT, V. C., TREDGET, E. E., CLANDININ, M. T. & FIELD, C. J. (2001) Fatty acid content of plasma lipids and erythrocyte phospholipids are altered following burn injury. *Lipids*, 36, 675-82.
- PRESSON, A. P., SOBEL, E. M., PAPP, J. C., SUAREZ, C. J., WHISTLER, T., RAJEEVAN, M. S., VERNON, S. D. & HORVATH, S. (2008) Integrated weighted gene co-expression network analysis with an application to chronic fatigue syndrome. *BMC Systems Biology*, 2, 95.
- PROUX, D., RECHENMANN, F., JULLIARD, L., PILLET, V. V. & JACQ, B. (1998) Detecting Gene Symbols and Names in Biological Texts: A First Step toward Pertinent Information Extraction. *Genome Inform Ser Workshop Genome Informics*, 9, 72-80.
- PUSTEJOVSKY, J., CASTANO, J., COCHRAN, B., KOTECKI, M. & MORRELL, M. (2001) Automatic extraction of acronym-meaning pairs from MEDLINE databases. *Medinfo*, 10, 371-5.
- PUSZTAI, L., AYERS, M., STEC, J., CLARK, E., HESS, K., STIVERS, D., DAMOKOSH, A., SNEIGE, N., BUCHHOLZ, T. A., ESTEVA, F. J., ARUN, B., CRISTOFANILLI, M., BOOSER, D., ROSALES, M., VALERO, V., ADAMS, C., HORTOBAGYI, G. N. & SYMMANS, W. F. (2003) Gene expression profiles obtained from fine-needle aspirations of breast cancer reliably identify routine prognostic markers and reveal large-scale molecular differences between estrogen-negative and estrogen-positive tumors. *Clinical Cancer Research*, 9, 2406-15.
- PYYSALO, S., AIROLA, A., HEIMONEN, J., BJORNE, J., GINTER, F. &

- SALAKOSKI, T. (2008) Comparative analysis of five protein-protein interaction corpora. *BMC Bioinformatics*, 9 Suppl 3, S6.
- QIU, P., GENTLES, A. J. & PLEVITIS, S. K. (2009) Fast calculation of pairwise mutual information for gene regulatory network reconstruction. *Computational Methods and Programs in Biomedicine*.
- QUIRK, S. J. & FUNDER, J. W. (1988) Steroid receptors, and the generation of closely coupled/biphasic dose-response curves. *Journal of Steroid Biochemistry*, 30, 9-15.
- QUIRK, S. J., GANNELL, J. E., FULLERTON, M. J. & FUNDER, J. W. (1986) Specificity and mechanism of biphasic action of glucocorticoids on alpha-lactalbumin production by rat mammary gland explants. *Endocrinology*, 118, 909-14.
- RATNAPARKHI, A. (1996) A Maximum Entropy Model for Part-of-Speech Tagging. IN BRILL, E. & CHURCH, K. (Eds.) *Conference on Empirical Methods in Natural Language Processing*. University of Pennsylvania, Association for Computational Linguistics.
- RAWOOL, S. B. & VENKATESH, K. V. (2007) Steady state approach to model gene regulatory networks-Simulation of microarray experiments. *Biosystems*.
- RAY, S. & CRAVEN, M. (2005) Learning statistical models for annotating proteins with function information using biomedical text. *BMC Bioinformatics*, 6 Suppl 1, S18.
- REVERTER, A., BARRIS, W., MORENO-SANCHEZ, N., MCWILLIAM, S., WANG, Y. H., HARPER, G. S., LEHNERT, S. A. & DALRYMPLE, B. P. (2005) Construction of gene interaction and regulatory networks in bovine skeletal muscle from expression data. *Australian Journal of Experimental Agriculture*, 45, 821-829.
- REYNAR, J. & RATNAPARKHI, A. (1997) A maximum entropy approach to identifying sentence boundaries. *Fifth Conference on Applied Natural Language Processing*. Washington, DC: University of Pennsylvania.
- RICCIO, P. (2004) The proteins of the milk fat globule membrane in the balance. *Trends in Food Science & Technology*, 15, 458-461.
- RINALDI, F., SCHNEIDER, G., KALJURAND, K., HESS, M., ANDRONIS, C., KONSTANDI, O. & PERSIDIS, A. (2007) Mining of relations between proteins over biomedical scientific literature using a deep-linguistic approach. *Artificial Intelligence in Medicine*, 39, 127-36.
- RINDFLESCH, T. C., RAJAN, J. & LAWRENCE HUNTER, L. (2000) Extracting molecular binding relationships from biomedical text. *6th Applied Natural*

- ROBERTS, P. M. (2006) Mining literature for systems biology. *Briefings in Bioinformatics*, 7, 399-406.
- ROBERTS, P. M. & HAYES, W. S. (2008) Information needs and the role of text mining in drug development. *Pacific Symposium of Biocomputing*, 592-603.
- ROBINSON, S. D., SILBERSTEIN, G. B., ROBERTS, A. B., FLANDERS, K. C. & DANIEL, C. W. (1991) Regulated expression and growth inhibitory effects of transforming growth factor-beta isoforms in mouse mammary gland development. *Development*, 113, 867-78.
- RODRIGUEZ-ESTEBAN, R., IOSSIFOV, I. & RZHETSKY, A. (2006) Imitating Manual Curation of Text-Mined Facts in Biomedicine. *PLoS Computational Biology*, 2.
- ROGERS, S. & GIROLAMI, M. (2005) A Bayesian regression approach to the inference of regulatory networks from gene expression data. *Bioinformatics*, 21, 3131-7.
- ROSEN, J. M., RODGERS, J. R., COUCH, C. H., BISBEE, C. A., DAVID-INOUE, Y., CAMPBELL, S. M. & YU-LEE, L. Y. (1986) Multihormonal regulation of milk protein gene expression. *Annals of the New York Academy of Science*, 478, 63-76.
- ROSMAN, A. S. & KORSTEN, M. A. (2007) Application of summary receiver operating characteristics (sROC) analysis to diagnostic clinical testing. *Advances in Medical Science*, 52, 76-82.
- ROUX, P. P. & BLENIS, J. (2004) ERK and p38 MAPK-activated protein kinases: a family of protein kinases with diverse biological functions. *Microbiology and Molecular Biology Reviews*, 68, 320-44.
- RUBINSTEIN, R. & SIMON, I. (2005) MILANO--custom annotation of microarray results using automatic literature searches. *BMC Bioinformatics*, 6, 12.
- RUDOLPH, M. C., MCMANAMAN, J. L., PHANG, T., RUSSELL, T., KOMINSKY, D. J., SERKOVA, N. J., STEIN, T., ANDERSON, S. M. & NEVILLE, M. C. (2007) Metabolic regulation in the lactating mammary gland: a lipid synthesizing machine. *Physiological Genomics*, 28, 323-336.
- RUSHTON, P. J., BOKOWIEC, M. T., LAUDEMAN, T. W., BRANNOCK, J. F., CHEN, X. & TIMKO, M. P. (2008) TOBFAC: the database of tobacco transcription factors. *BMC Bioinformatics*, 9, 53.
- RYAN, T. M., WINTERS, R. S., MARK, M., YANG, J., PETER, S. W. & FERNANDO, P. (2004) An entity tagger for recognizing acquired genomic

- variations in cancer literature. *Bioinformatics*, 20, 3249.
- RZHETSKY, A., IOSSIFOV, I., KOIKE, T., KRAUTHAMMER, M., KRA, P., MORRIS, M., YU, H., DUBOUE, P. A., WENG, W., WILBUR, W. J., HATZIVASSILOGLU, V. & FRIEDMAN, C. (2004) GeneWays: a system for extracting, analyzing, visualizing, and integrating molecular pathway data. *Journal of Biomedical Informatics*, 37, 43-53.
- RZHETSKY, A., ZHENG, T. & WEINREB, C. (2006) Self-correcting maps of molecular pathways. *PLoS ONE*, 1, e61.
- SARKAR, I. N. & AGRAWAL, A. (2006) Literature based discovery of gene clusters using phylogenetic methods. *Proceedings of the AMIA Annual Symposium*, 689-93.
- SAKHARKAR, M. K. & KANGUEANE, P. (2004) Genome SEGE: a database for 'intronless' genes in eukaryotic genomes. *BMC Bioinformatics*, 5, 67.
- SAKLATVALA, J. (2004) The p38 MAP kinase pathway as a therapeutic target in inflammatory disease. *Current Opinions in Pharmacology*, 4, 372-7.
- SALWINSKI, L., MILLER, C. S., SMITH, A. J., PETTIT, F. K., BOWIE, J. U. & EISENBERG, D. (2004) The Database of Interacting Proteins: 2004 update. *Nucleic Acids Research*, 32, D449-51.
- SANTOS, C., EGGLE, D. & STATES, D. J. (2005) Wnt pathway curation using automated natural language processing: combining statistical methods with partial and full parse for knowledge extraction. *Bioinformatics*, 21, 1653-8.
- SAKAKURA, T., KUSANO, I., KUSAKABE, M., INAGUMA, Y. & NISHIZUKA, Y. (1987) Biology of mammary fat pad in fetal mouse: capacity to support development of various fetal epithelia in vivo. *Development*, 100, 421-30.
- SARIC, J., JENSEN, L. J., OUZOUNOVA, R., ROJAS, I. & BORK, P. (2005) Extraction of regulatory gene/protein networks from Medline. *Bioinformatics*.
- SASIK, R., HWA, T., IRANFAR, N. & LOOMIS, W. F. (2001) Percolation clustering: a novel algorithm applied to the clustering of gene expression patterns in dictyostelium development. *Proceedings of the Pacific Symposium of Biocomputing*, 6, 335-347.
- SATTERTHWAITE, F. E. (1942) Generalized Poisson Distribution. *The Annals of Mathematical Statistics*, 13, 410-417.
- SCHADT, E. E., LI, C., ELLIS, B. & WONG, W. H. (2001) Feature extraction and normalization algorithms for high-density oligonucleotide gene expression array data. *Journal of Cellular Biochemistry*, 84, 120-125.



- SCHNEIDER, G., RINALDI, F. & DOWDALL, J. (2004) Fast, deep-linguistic statistical dependency parsing. *20th International Conference on Computational Linguistics*. University of Geneva, Switzerland, Association of Computational Linguistics.
- SCHROEDER, J. A. & LEE, D. C. (1997) Transgenic mice reveal roles for TGF $\alpha$  and EGF receptor in mammary gland development and neoplasia. *Journal of Mammary Gland Biology and Neoplasia*, 2, 119-29.
- SCHWARTZ, A. S. & HEARST, M. A. (2003) A simple algorithm for identifying abbreviation definitions in biomedical text. *Proceedings of the Pacific Symposium on Biocomputing*, 8, 451-462.
- SCMIDT, G. H. (1966) Effect of insulin on yield and composition of milk of dairy cows. *Journal of Dairy Science*, 47, 381-385.
- SCOTT, M., LU, G., HALLETT, M. & THOMAS, D. Y. (2004) The Hera database and its use in the characterization of endoplasmic reticulum proteins. *Bioinformatics*, 20, 937-44.
- SEBOLT-LEOPOLD, J. S. (2000) Development of anticancer drugs targeting the MAP kinase pathway. *Oncogene*, 19, 6594-9.
- SETTLES, B. (2005) ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21, 3191-2.
- SHAH, P. K., JENSEN, L. J., BOUE, S. & BORK, P. (2005) Extraction of transcript diversity from scientific literature. *PLoS Computational Biology*, 1, e10.
- SHANNON, P., MARKIEL, A., OZIER, O., BALIGA, N. S., WANG, J. T., RAMAGE, D., AMIN, N., SCHWIKOWSKI, B. & IDEKER, T. (2003) Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13, 2498-504.
- SHATKAY, H. & WILBUR, W. J. (2000) Finding themes in Medline documents: probabilistic similarity search. *IEEE Conference on Advances in Digital Libraries*.
- SHATKAY, H., PAN, F., RZHETSKY, A. & WILBUR, W. J. (2008) Multi-dimensional classification of biomedical text: toward automated, practical provision of high-utility text to diverse users. *Bioinformatics*, 24, 2086-93.
- SHERLOCK, G., HERNANADEZ-BOUSSARD, T., KASARSKIS, A., BINKLEY, G., MATESE, J. C., DWIGHT, S. S., KALOPER, M., WENG, S., JIN, H., BALL, C. A., EISEN, M. B., SPELLMAN, P. T., BROWN, P. O., BOTSTEIN, D. & CHERRY, J. M. (2001) The Stanford microarray database. *Nucleic Acid Research*, 29, 152-155.

- SHI, L. & CAMPAGNE, F. (2005) Building a protein name dictionary from full text: a machine learning term extraction approach. *BMC Bioinformatics*, 6, 88.
- SHIMAMURA, T., IMOTO, S., YAMAGUCHI, R. & MIYANO, S. (2007) Weighted lasso in graphical Gaussian modeling for large gene network estimation based on microarray data. *Genome Informatics*, 19, 142-53.
- SIMON, M. L., PATRICK, M., KIMBERLY, F. J. & JENNIFER, S. (2004) MedlineR: an open source library in R for Medline literature data mining. *Bioinformatics*, 20, 3659.
- SINGH, M. K., SRIVASTAVA, S., RAGHAVA, G. P. & VARSHNEY, G. C. (2006) HaptenDB: a comprehensive database of haptens, carrier proteins and anti-hapten antibodies. *Bioinformatics*, 22, 253-5.
- SINHA, Y. N., SELBY, F. W. & VANDERLAAN, W. P. (1974) Relationship of prolactin and growth hormone to mammary function during pregnancy and lactation in the C3H-ST mouse. *Journal of Endocrinology*, 61, 219-29.
- SKUSA, A., RUEGG, A. & KOHLER, J. (2005) Extraction of biological interaction networks from scientific literature. *Briefings in Bioinformatics*, 6, 263-76.
- SLEATOR, D. & TEMPERLEY, D. (1991) Parsing English with a Link Grammar. *Third International Workshop on Parsing Technologies*.
- SLEBODZINSKI, A. B., BRZEZINSKA-SLEBODZINSKA, E., STYCZYNSKA, E. & SZEJNOGA, M. (1999) Presence of thyroxine deiodinases in mammary gland: possible modulation of the enzyme-deiodinating activity by somatotropin. *Domestic Animal Endocrinology*, 17, 161-9.
- SMALHEISER, N. R., TORVIK, V. I. & ZHOU, W. (2009) Arrowsmith two-node search interface: A tutorial on finding meaningful links between two disparate sets of articles in MEDLINE. *Computer Methods and Programs in Biomedicine*, 94, 190-7.
- SMITH, L., RINDFLESCH, T. & WILBUR, W. J. (2004) MedPost: a part-of-speech tagger for bioMedical text. *Bioinformatics*, 20, 2320-1.
- SOHN, S., COMEAU, D., KIM, W. & WILBUR, W. J. (2008) Abbreviation definition identification based on automatic precision estimates. *BMC Bioinformatics*, 9, 402.
- SONG, J. H., KIM, J. M., KIM, S. H., KIM, H. J., LEE, J. J., SUNG, M. H., HWANG, S. Y. & KIM, T. S. (2003) Comparison of the gene expression profiles of monocytic versus granulocytic lineages of HL-60 leukemia cell differentiation by DNA microarray analysis. *Life Science*, 73, 1705-19.
- SPASIC, I., SCHOBBER, D., SANSONE, S. A., REBHOLZ-SCHUHMANN, D., KELL,

- D. B. & PATON, N. W. (2008) Facilitating the development of controlled vocabularies for metabolomics technologies with text mining. *BMC Bioinformatics*, 9 Suppl 5, S5.
- SRINIVASAN, P., LIBBUS, B. & SEHGAL, A. K. (2004) Mining MEDLINE: postulating a beneficial role for Curcumin Longa in retinal diseases. *BioLink 2004: Linking Biological Literature, Ontologies, and Databases*. Association for Computational Linguistics.
- STAPLEY, B. J. & BENOIT, G. (2000) Biobibliometrics: information retrieval and visualization from co-occurrences of gene names in medline abstracts. *Proceedings of the Pacific Symposium on Biocomputing*, 5, 526-537.
- STEPHENS, M., APLAKAL, M., MUKHOPADHYAY, S., RAJE, R. & MOSTAFA, J. (2001) Detecting gene relations from MEDLINE abstracts. *Proceedings of the Pacific Symposium on Biocomputing*, 6, 483-496.
- STEIN, T., MORRIS, J., DAVIES, C., WEBER-HALL, S., DUFFY, M.-A., HEATH, V., BELL, A., FERRIER, R., SANDILANDS, G. & GUSTERSON, B. (2004) Involution of the mouse mammary gland is associated with an immune cascade and an acute-phase response, involving LBP, CD14 and STAT3. *Breast Cancer Research*, 6, R75 - R91.
- STOLOVITZKY, G. A., KUNDAJE, A., HELD, G. A., DUGGAR, K. H., HAUDENSCHILD, C. D., ZHOU, D., VASICEK, T. J., SMITH, K. D., ADEREM, A. & ROACH, J. C. (2005) Statistical analysis of MPSS measurements: application to the study of LPS-activated macrophage gene expression. *Proceedings of the National Academy of Science U S A*, 102, 1402-7.
- STUART, J. M., SEGAL, E., KOLLER, D. & KIM, S. K. (2003) A gene-coexpression network for global discovery of conserved genetic modules. *Science*, 302, 249-55.
- SWANSON, D. R. (1986) Fish oil, Raynaud's syndrome, and undiscovered public knowledge. *Perspectives in Biology and Medicine*, 30, 7-18.
- SWANSON, D. R. (1988) Migraine and magnesium: eleven neglected connections. *Perspectives in Biology and Medicine*, 31, 526-57.
- SWANSON, D. R. (1990a) Medical literature as a potential source of new knowledge. *Bulletin of the Medical Library Association*, 78, 29-37.
- SWANSON, D. R. (1990b) Somatomedin C and arginine: implicit connections between mutually isolated literatures. *Perspectives in Biology and Medicine*, 33, 157-86.
- SWETS, J. A. (1988) Measuring the accuracy of diagnostic systems. *Science*, 240, 1285-93.

- TABIBIAZAR, R., WAGNER, R. A., LIAO, A. & QUERTERMOUS, T. (2003) Transcriptional profiling of the heart reveals chamber-specific gene expression patterns. *Circulation Research*, 93, 1193-201.
- TAIB, Z. (2003) New estimates of gene expression for oligonucleotide microchip arrays. *Comptes Rendus de l'Academie des Sciences B*.
- TAHA, C. & KLIP, A. (1999) The insulin signaling pathway. *Journal of Membrane Biology*, 169, 1-12.
- TAMAYO, P., SLONIM, D., MESIROV, J., ZHU, Q., KITAREEWAN, S., DMITROVSKY, E., LANDER, E. S. & GOLUB, T. R. (1999) Interpreting patterns of gene expression with self-organizing maps: methods and applications to hematopoietic differentiation. *Proceedings of the Pacific Symposium of Biocomputing*, 4, 5-16.
- TANABE, L., SCHERF, U., SMITH, L. H., LEE, J. K., HUNTER, L. & WEINSTEIN, J. N. (1999) MedMiner: an Internet text-mining tool for biomedical information, with application to gene expression profiling. *Biotechniques*, 27, 1210-4, 1216-7.
- TANABE, L. & WILBUR, W. J. (2002) Tagging gene and protein names in biomedical text. *Bioinformatics*, 18, 1124-32.
- TANABE, L., XIE, N., THOM, L. H., MATTEN, W. & WILBUR, W. J. (2005) GENETAG: a tagged corpus for gene/protein named entity recognition. *BMC Bioinformatics*, 6 Suppl 1, S3.
- TATARCZUCH, L., PHILIP, C. & LEE, C. S. (1997) Involution of the sheep mammary gland. *Journal of Anatomy*, 190 (Pt 3), 405-16.
- TATEISI, Y. & TSUJI, J. I. (2004) Part-of-Speech Annotation of Biology Research Abstracts. *4th International Conference on Language Resource and Evaluation (LREC2004)*.
- TEMKIN, J. M. & GILDER, M. R. (2003) Extraction of protein interaction information from unstructured text using a context-free grammar. *Bioinformatics*, 19, 2046-53.
- TENG, C. T., PENTECOST, B. T., CHEN, Y. H., NEWBOLD, R. R., EDDY, E. M. & MCLACHLAN, J. A. (1989) Lactotransferrin gene expression in the mouse uterus and mammary gland. *Endocrinology*, 124, 992-9.
- TERRY, P. M., BANERJEE, M. R. & LUI, R. M. (1977) Hormone-inducible casein messenger RNA in a serum-free organ culture of whole mammary gland. *Proceedings of the National Academy of Science U S A*, 74, 2441-5.

- THEODOROPOULOU, M. C., BAGOS, P. G., SPYROPOULOS, I. C. & HAMODRAKAS, S. J. (2008) gpDB: a database of GPCRs, G-proteins, effectors and their interactions. *Bioinformatics*, 24, 1471-2.
- THOMAS, R., MEHROTRA, S., PAPOUTSAKIS, E. T. & HATZIMANIKATIS, V. (2004) A model-based optimization framework for the inference on gene regulatory networks from DNA array data. *Bioinformatics*, 20, 3221-35.
- TIFFIN, N., KELSO, J. F., POWELL, A. R., PAN, H., BAJIC, V. B. & HIDE, W. A. (2005) Integration of text- and data-mining using ontologies successfully selects disease gene candidates. *Nucleic Acids Research*, 33, 1544-52.
- TOH, H. & HORIMOTO, K. (2002) Inference of a genetic network by a combined approach of cluster analysis and graphical Gaussian modeling. *Bioinformatics*, 18, 287-97.
- TORII, M., HU, Z. Z., SONG, M., WU, C. H. & LIU, H. (2007) A comparison study on algorithms of detecting long forms for short forms in biomedical text. *BMC Bioinformatics*, 8 Suppl 9, S5.
- TOPINKA, C. & SHYU, C. R. (2006) Predicting cancer interaction networks using text-mining and structure understanding. *Proceedings of the AMIA Annual Symposium*, 1123.
- TOPPER, Y. J. & FREEMAN, C. S. (1980) Multiple hormone interactions in the developmental biology of the mammary gland. *Physiological Reviews*, 60, 1049-1106.
- TOPPER, Y. J., NICHOLAS, K. R., SANKARAN, L. & KULSKI, J. (1984a) Insulin as a developmental hormone. *Hormones and Cancer*.
- TOPPER, Y. J., NICHOLAS, K. R., SANKARAN, L. & KULSKI, J. K. (1984b) Insulin biology from the perspective of studies on mammary gland development. *Biochemical actions of hormone*. Academic Press, Inc.
- TUOYA, SUGII, Y., SATOH, H., YU, D., MATSUURA, Y., TOKUTAKA, H. & SENO, M. (2008) Spherical self-organizing map as a helpful tool to identify category-specific cell surface markers. *Biochemical and Biophysical Research Communications*, 376, 414-8.
- TURKINGTON, R. W., JUERGENS, W. G. & TOPPER, Y. J. (1965) Hormone-dependent synthesis of casein in vitro. *Biochimica et Biophysica Acta*, 111, 573-6.
- TURKINGTON, R. W., LOCKWOOD, D. H. & TOPPER, Y. J. (1967) The induction of milk protein synthesis in post-mitotic mammary epithelial cells exposed to prolactin. *Biochimica et Biophysica Acta*, 148, 475-80.

- TSAI, R. T.-H., WU, S.-H., CHOU, W.-C., LIN, Y.-C., HE, D., HSIANG, J., SUNG, T.-Y. & HSU, W.-L. (2006) Various criteria in the evaluation of biomedical named entity recognition. *BMC Bioinformatics*, 7, 92.
- TSENG, G. C., OH, M.-K., ROHLIN, L., LIAO, J. C. & WONG, W. H. (2001) Issues in cDNA microarray analysis: quality filtering, channel normalization, models of variation and assessment of gene effects. *Nucleic Acid Research*, 29, 2549-2557.
- TSURUOKA, Y. & TSUJII, J. (2004) Improving the performance of dictionary-based approaches in protein name recognition. *Journal of Biomedical Informatics*, 37, 461-70.
- ULITSKY, I. & SHAMIR, R. (2007) Identification of functional modules using network topology and high-throughput data. *BMC System Biology*, 1, 8.
- URAMOTO, N., MATSUZAWA, H., NAGANO, T., MURAKAMI, A., TAKEUCHI, H. & TAKEDA, K. (2004) A text-mining system for knowledge discovery from biomedical documents. *IBM Systems Journal*, 43, 516-533.
- VAISHNAV, D., JAMBAL, P., REUSCH, J. E. & PUGAZHENTHI, S. (2003) SP600125, an inhibitor of c-jun N-terminal kinase, activates CREB by a p38 MAPK-mediated pathway. *Biochemical and Biophysical Research Communication*, 307, 855-60.
- VAN ECK, N. J. (2005) Towards automatic knowledge discovery from scientific literature. *Econometric Institute, Faculty of Economics*. Rotterdam, Erasmus University.
- VAN ECK, N. J. & VAN DEN BERG, J. (2005) A novel algorithm for visualizing concept associations. *16th International Workshop on Database and Expert System Applications (DEXA'05)*.
- VAZQUEZ-MARTINEZ, R., CRUZ-GARCIA, D., DURAN-PRADO, M., PEINADO, J. R., CASTANO, J. P. & MALAGON, M. M. (2007) Rab18 inhibits secretory activity in neuroendocrine cells by interacting with secretory granules. *Traffic*, 8, 867-82.
- VENUGOPAL, S., BUYYA, R. & RAMAMOHANARAO, K. (2006) A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computing Surveys*, 38, Article No. 3.
- VINCZE, V., SZARVAS, G., FARKAS, R., MORA, G. & CSIRIK, J. (2008) The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9 Suppl 11, S9.
- VLASBLOM, J., WU, S., PU, S., SUPERINA, M., LIU, G., ORSI, C. & WODAK, S. J. (2006) GenePro: a Cytoscape plug-in for advanced visualization and analysis of

- interaction networks. *Bioinformatics*, 22, 2178-9.
- VON MERING, C., JENSEN, L. J., SNEL, B., HOOPER, S. D., KRUPP, M., FOGLIERINI, M., JOUFFRE, N., HUYNEN, M. A. & BORK, P. (2005) STRING: known and predicted protein-protein associations, integrated and transferred across organisms. *Nucleic Acids Research*, 33, D433-7.
- VONDERHAAR, B. K. (1977) Studies on the mechanism by which thyroid hormones enhance alpha-lactalbumin activity in explants from mouse mammary glands. *Endocrinology*, 100, 1423-31.
- VONDERHAAR, B. K. (1988) Regulation of development of the normal mammary gland by hormones and growth factors. *Cancer Treatment Research*, 40, 251-66.
- VONDERHAAR, B. K. & SMITH, G. H. (1982) Dissociation of cytological and functional differential in virgin mouse mammary gland during inhibition of DNA synthesis. *Journal of Cell Science*, 53, 97-114.
- VOORHEES, E. & BUCKLAND, L. P. (Eds.) (2005) *The Fourteen Text REtrieval Conference Proceedings*, Gaithersburg, Maryland, National Institute of Standards and Technology (NIST), the Defense Advanced Research Projects Agency (DARPA) and the Advanced Research and Development Activity (ARDA).
- VUCETIC, S., OBRADOVIC, Z., VACIC, V., RADIVOJAC, P., PENG, K., IAKOUCHEVA, L. M., CORTESE, M. S., LAWSON, J. D., BROWN, C. J., SIKES, J. G., NEWTON, C. D. & DUNKER, A. K. (2005) DisProt: a database of protein disorder. *Bioinformatics*, 21, 137-40.
- WALDEN, P. D., RUAN, W., FELDMAN, M. & KLEINBERG, D. L. (1998) Evidence that the mammary fat pad mediates the action of growth hormone in mammary gland development. *Endocrinology*, 139, 659-662.
- WALL, M. E., DYCK, P. A. & BRETTIN, T. S. (2001) SVDMAN - singular value decomposition analysis of microarray data. *Bioinformatics*, 17, 566-568.
- WALL, E. H. & MCFADDEN, T. B. (2008) Use it or lose it: enhancing milk production efficiency by frequent milking of dairy cows. *Journal of Animal Science*, 86, 27-36.
- WANG, Y., JOSHI, T., ZHANG, X. S., XU, D. & CHEN, L. (2006) Inferring gene regulatory networks from multiple microarray datasets. *Bioinformatics*, 22, 2413-20.
- WANG, H. & NUTTALL, P. A. (1999) Immunoglobulin-binding proteins in ticks: new target for vaccine development against a blood-feeding parasite. *Cellular and Molecular Life Sci*, 56, 286-95.

- WEAVER, D. C., WORKMAN, C. T. & STORMO, G. D. (1999) Modeling regulatory networks with weight matrices. *Proceedings of the Pacific Symposium of Biocomputing*, 4, 112-123.
- WEEBER, M., KORS, J. A. & MONS, B. (2005) Online tools to support literature-based discovery in the life sciences. *Briefings in Bioinformatics*, 6, 277-86.
- WEINSTEIN, D., BEN-DAVID, M. & POLISHUK, W. Z. (1976) Serum prolactin and the suppression of lactation. *British Journal of Obstetrics and Gynaecology*, 83, 679-82.
- WERHLI, A. V. & HUSMEIER, D. (2007) Reconstructing gene regulatory networks with bayesian networks by combining expression data with multiple sources of prior knowledge. *Statistical Applications in Genetics and Molecular Biology*, 6, Article15.
- WHEELER, D. L., BARRETT, T., BENSON, D. A., BRYANT, S. H., CANESE, K., CHETVERNIN, V., CHURCH, D. M., DICUCCIO, M., EDGAR, R., FEDERHEN, S., GEER, L. Y., HELMBERG, W., KAPUSTIN, Y., KENTON, D. L., KHOVAYKO, O., LIPMAN, D. J., MADDEN, T. L., MAGLOTT, D. R., OSTELL, J., PRUITT, K. D., SCHULER, G. D., SCHRIML, L. M., SEQUEIRA, E., SHERRY, S. T., SIROTKIN, K., SOUVOROV, A., STARCHENKO, G., SUZEK, T. O., TATUSOV, R., TATUSOVA, T. A., WAGNER, L. & YASCHENKO, E. (2006) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 34, D173-80.
- WHETZEL, P. L., PARKINSON, H. & STOECKERT, C. J., JR. (2006) Using ontologies to annotate microarray experiments. *Methods in Enzymology*, 411, 325-39.
- WILBUR, W. J. (2002) A thematic analysis of the AIDS literature. *Proceedings of the Pacific Symposium on Biocomputing*, 7, 386-397.
- WILBUR, W. J., RZHETSKY, A. & SHATKAY, H. (2006) New directions in biomedical text annotation: definitions, guidelines and corpus construction. *BMC Bioinformatics*, 7, 356.
- WILBUR, W. J. & YANG, Y. (1996) An analysis of statistical term strength and its use in the indexing and retrieval of molecular biology texts. *Computers in Biology & Medicine*, 26, 209-22.
- WINDER, S. J. & AYSCOUGH, K. R. (2005) Actin-binding proteins. *Journal of Cell Science*, 118, 651-4.
- WINTER, E. E., GOODSTADT, L. & PONTING, C. P. (2004) Elevated rates of protein secretion, evolution, and disease among tissue-specific genes. *Genome Research*, 14, 54-61.



- WITTE, R., KAPPLER, T. & BAKER, C. J. (2007) Enhanced semantic access to the protein engineering literature using ontologies populated by text mining. *International Journal of Bioinformatics Research and Applications*, 3, 389-413.
- WONG, L. (2001) PIES, a protein interaction extraction system. *Pacific Symposium on Biocomputing*, 520-31.
- WREN, J. D., CHANG, J. T., PUSTEJOVSKY, J., ADAR, E., GARNER, H. R. & ALTMAN, R. B. (2005) Biomedical term mapping databases. *Nucleic Acids Research*, 33, D289-93.
- WREN, J. D. & GARNER, H. R. (2002) Heuristics for identification of acronym-definition patterns within text: towards an automated construction of comprehensive acronym-definition dictionaries. *Methods of Information in Medicine*, 41, 426-34.
- WREN, J. D. & GARNER, H. R. (2004) Shared relationship analysis: ranking set cohesion and commonalities within a literature-derived relationship network. *Bioinformatics*, 20, 191-8.
- WU, C. C., HUANG, H. C., JUAN, H. F. & CHEN, S. T. (2004) GeneNetwork: an interactive tool for reconstruction of genetic networks using microarray data. *Bioinformatics*, 20, 3691-3.
- WU, G., NIE, L. & FREELAND, S. J. (2007) The effects of differential gene expression on coding sequence features: analysis by one-way ANOVA. *Biochemical and Biophysical Research Communications*, 358, 1108-13.
- WYSS, C. M. & ROBERTSON, E. L. (2005) Relational languages for metadata integration. *ACM Transactions on Database Systems*, 30, 624-660.
- XING, E. P. & KARP, R. M. (2001) CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. *Proceedings of the International Society of Molecular Biology*.
- XU, H., ANDERSON, K., GRANN, V. & FRIEDMAN, C. (2004a) Facilitating cancer research using natural language processing of pathological reports. *11th World Congress on Medical Informatics*. San Francisco, CA, IMIA.
- XU, Q. G., LI, X. Q., KOTACHA, S. A., CHENG, C., SUN, H. S. & ZOCHODNE, D. W. (2004b) Insulin as an in vivo growth factor. *Experimental Neurology*, 188, 43-51.
- XU, Y., WANG, Z., LEI, Y., ZHAO, Y. & XUE, Y. (2009) MBA: a literature mining system for extracting biomedical abbreviations. *BMC Bioinformatics*, 10, 14.
- YAKUSHIJI, A., TATEISI, Y., MIYAO, Y. & TSUJII, J. (2001) Event extraction from biomedical papers using a full parser. *Proceedings of the Pacific Symposium on*

- YAMASHITA, H., NEVALAINEN, M. T., XU, J., LEBARON, M. J., WAGNER, K.-U., ERWIN, R. A., HARMON, J. M., HENNIGHAUSEN, L., KIRKEN, R. A. & RUI, H. (2001) Role of serine phosphorylation of Stat5a in prolactin-stimulated b-casein gene expression. *Molecular and Cellular Endocrinology*, 183, 151–163.
- YANG, C. Y., CHANG, C. H., YU, Y. L., LIN, T. C., LEE, S. A., YEN, C. C., YANG, J. M., LAI, J. M., HONG, Y. R., TSENG, T. L., CHAO, K. M. & HUANG, C. Y. (2008a) PhosphoPOINT: a comprehensive human kinase interactome and phospho-protein database. *Bioinformatics*, 24, i14-20.
- YANG, J., COHEN, A. & HERSH, W. (2009) Evaluation of a gene information summarization system by users during the analysis process of microarray datasets. *BMC Bioinformatics*, 10 Suppl 2, S5.
- YANG, H., NENADIC, G. & KEANE, J. A. (2008b) Identification of transcription factor contexts in literature using machine learning approaches. *BMC Bioinformatics*, 9 Suppl 3, S11.
- YANG, Y. H., DUDOIT, S., LUU, P., LIN, D. M., PENG, V., NGAI, J. & SPEED, T. P. (2002) Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research*, 30, e15.
- YETISGEN-YILDIZ, M. & PRATT, W. (2006) Using statistical and knowledge-based approaches for literature-based discovery. *Journal of Biomedical Informatics*, 39, 600-11.
- YIMENG, D., PIERRE-FRANÇOIS, B., GIANLUCA, P., YANN, P., JAMES, N. & PIERRE, B. (2004) ICBS: a database of interactions between protein chains mediated by  $\beta$ -sheet formation. *Bioinformatics*, 20, 2767.
- YOO, I., HU, X. & SONG, I. Y. (2007) Biomedical ontology improves biomedical literature clustering performance: a comparison study. *International Journal of Bioinformatics Research and Applications*, 3, 414-28.
- YOSHIMURA, M. & OKA, T. (1990) Hormonal induction of b-casein gene expression: requirement of ongoing protein synthesis for transcription. *Endocrinology*, 126, 427-433.
- YU, H. & AGICHTEIN, E. (2003) Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, 19 Suppl 1, i340-9.
- YU, H., HATZIVASSILOGLOU, V., FRIEDMAN, C., RZHETSKY, A. & WILBUR, W. J. (2002a) Automatic extraction of gene and protein synonyms from

MEDLINE and journal articles. *Proceedings of the AMIA Annual Symposium*.

YU, H., HRIPCSAK, G. & FRIEDMAN, C. (2002b) Mapping abbreviations to full forms in biomedical articles. *Journal of the American Medical Informatics Association*, 9, 262-72.

YU, L., AHMED, S. T., GONZALEZ, G., LOGSDON, B., NAKAMURA, M., NIKKILA, S., SHAH, K., TARI, L., WENDT, R., ZEIGLER, A. & BARAL, C. (2005) Genomic information retrieval through selective extraction and tagging by the ASU-BioAI group. *14th Text Retrieval Conference (TREC2005)*.

YURYEV, A., MULYUKOV, Z., KOTELNIKOVA, E., MASLOV, S., EGOROV, S., NIKITIN, A., DARASELIA, N. & MAZO, I. (2006) Automatic pathway building in biological association networks. *BMC Bioinformatics*, 7, 171.

ZAKARIA, Z. A., MAT JAIS, A. M., GOH, Y. M., SULAIMAN, M. R. & SOMCHIT, M. N. (2007) Amino acid and fatty acid composition of an aqueous extract of *Channa striatus* (Haruan) that exhibits antinociceptive activity. *Clinical and Experimental Pharmacology and Physiology*, 34, 198-204.

ZARUBIN, T. & HAN, J. (2005) Activation and signaling of the p38 MAP kinase pathway. *Cell Research*, 15, 11-8.

ZHANG, S., XIA, X., SHEN, J., ZHOU, Y. & SUN, Z. (2008) DBMLoc: a Database of proteins with multiple subcellular localizations. *BMC Bioinformatics*, 9, 127.

ZHANG, G.-Q., TROY, A. D. & BOURGOIN, K. (2006a) Bootstrapping ontology learning for information retrieval using formal concept analysis and information anchors. *14th International Conference on Conceptual Structures*. Aalborg, Denmark.

ZHANG, W., YU, C., SMALHEISER, N. & TORVIK, V. (2006b) Segmentation of publication records of authors from the web. *22nd IEEE International Conference on Data Engineering (ICDE'06)*. Atlanta, Georgia, IEEE Press.

ZHANG, Z., TANG, S. & NG, S. (2005) Toward discovering disease-specific gene networks from online literature. IN CHEN, Y.-P. P. & WONG, L. (Eds.) *The 3rd Asia-Pacific Bioinformatics Conference*. Singapore, Imperial College Press.

ZHENG, C. J., ZHOU, H., XIE, B., HAN, L. Y., YAP, C. W. & CHEN, Y. Z. (2004) TRMP: a database of therapeutically relevant multiple pathways. *Bioinformatics*, 20, 2236-41.

ZHAO, F.-Q., ADACHI, K. & OKA, T. (2002) Involvement of Oct-1 in transcriptional regulation of b-casein gene expression in mouse mammary gland. *Biochemica et Biophysica Acta*, 1577, 27-37.

- ZHAO, W., SERPEDIN, E. & DOUGHERTY, E. R. (2008) Inferring connectivity of genetic regulatory networks using information-theoretic criteria. *IEEE/ACM Transactions in Computational Biology and Bioinformatics*, 5, 262-74.
- ZHOU, Y. & ABAGYAN, R. (2002) Match-only intergral distribution (MOID) algorithm for high-density oligonucleotide array analysis. *BMC Bioinformatics*, 3, 3.
- ZHOU, D. & HE, Y. (2008) Extracting interactions between proteins from the literature. *Journal of Biomedical Informatics*, 41, 393-407.
- ZHOU, G., SHEN, D., ZHANG, J., SU, J. & TAN, S. (2005) Recognition of protein/gene names from text using an ensemble of classifiers. *BMC Bioinformatics*, 6 Suppl 1, S7.
- ZHOU, W., SMALHEISER, N. R. & YU, C. (2006) A tutorial on information retrieval: basic terms and concepts. *Journal of Biomedical Discovery and Collaboration*, 1, 2.
- ZHOU, W., TORVIK, V. I. & SMALHEISER, N. R. (2006b) ADAM: another database of abbreviations in MEDLINE. *Bioinformatics*, 22, 2813-8.
- ZHU, Y., SHEN, X. & PAN, W. (2009) Network-based support vector machine for classification of microarray samples. *BMC Bioinformatics*, 10 Suppl 1, S21.
- ZIEN, A., KUFFNER, R., ZIMMER, R. & LENGAUER, T. (2000) Analysis of gene expression data with pathway scores. *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (AAAI Press)*.
- ZISKA, S. E., BHATTACHARJEE, M., HERBER, R. L., QASBA, P. K. & VONDERHAAR, B. K. (1988) Thyroid hormone regulation of alpha-lactalbumin: differential glycosylation and messenger ribonucleic acid synthesis in mouse mammary glands. *Endocrinology*, 123, 2242-8.
- ZOU, M. & CONZEN, S. D. (2005) A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21, 71-9.

## Appendix A

### Selected Source Codes

#### ***A.1. Introduction***

This appendix lists selected parts of Muscorian and its related tools (about 4600 lines of codes) as used in this thesis to illustrate the main operations of Muscorian. Although the operations of Muscorian was described in Chapter 2, Muscorian had been extended to include statistical co-occurrence method of biomedical literature analysis described in Chapter 4, as well as microarray data and analyses described in Chapters 5. The entire codebase contained about 600000 lines of codes and comprised of the following external modules:

1. MontyLingua 2.1 by Hugo Liu, MIT Media Labs (under General Public Licence)
2. codes from ASPN Python Cookbook Recipe 81330 by Xavier Defrang and alane@sourceforge.net (<http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/81330>)
3. codes from 'Simplified Exception Identification in Python' by Jean-Francois Touchette. On Linux Journal, July 28, 2002 (<http://www.linuxjournal.com/article.php?sid=5821>)
4. BioNLP Web Services by Jeffrey T. Chang <[jeffrey.chang@duke.edu](mailto:jeffrey.chang@duke.edu)> (PhD thesis of Jeffrey T. Chang at Stanford University)
5. MedPost (L. Smith, T. Rindflesch and W. J. Wilbur. 2004. MedPost: a part-of-speech tagger for bioMedical text. *Bioinformatics* 20(14):2320-2321; doi:10.1093/bioinformatics/bth227)
6. Textpresso ontology (H.M. Muller, E.E. Kenny and P.W. Sternberg. Textpresso: an ontology-based information retrieval and extraction system for biological literature. *PLoS Biology* 2(11):e309; doi:10.1371/journal.pbio.0020309)
7. Natural Language Toolkit ([nltk.sourceforge.net](http://nltk.sourceforge.net))
8. codes from <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/190465>
9. IDCross (<http://ib-dwb.sourceforge.net/IDCross>) as mureus
10. pyparsing (<http://pyparsing.sourceforge.net/>)

11. pydot (<http://dkbza.org/pydot.html>)
12. codes for sampling with replacement (<http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/273085>)

## ***A.2. High-Level Description of the Codes***

Five code files from Muscorian (Section A.3) and one code file containing analysis of incremental stepwise mapping of Masters et al. (2002)'s co-expression network to 1-PubGene co-occurrence network (Chapter 4) are listed. The purpose of each of the five code files from Muscorian are as follows:

File 'muscopedia.sql' contains the SQL codes for the definition of Muscopedia, the Firebird database for storing raw text data, extracted protein-protein interactions and its intermediate forms of each stage of natural language processing, as described in Section 2.3.1. In addition to natural language processing, Muscopedia is also extended to cater for other storage aspects of this thesis, such as co-occurrence calculations (Chapter 5), microarray data sets and co-expression networks (Chapter 5).

File 'muscorian.py' is the entry point and driver of Muscorian. It provides a command-line interface to all the operations used in this thesis. In short, it brings together all the components of Muscorian to the user.

File 'UCFunctions.py' contains codes to filter MontyLingua's subject-verb-object (SVO) output, such as removing incomplete SVOs and adjoining multiple objects phrases into a single object phrase. It also contains short “borrowed” modules from other sources, such as codes for sampling with replacement (<http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/273085>). Essentially, 'UCFunctions' means unclassified functions or functions that do not fit into any modules.

File 'TextReplacer.py' contains the entity normalization codes as described in Section 2.2.1. Its main operation is to prepare the dictionary of curated abbreviations and using it to entity normalize each of the abstracts given.

File 'PubMedGrabber.py' contains the codes to retrieve citations from PubMed and loading it into Musclopedia. At the same time, it also contains the codes to connect to Musclopedia using Firebird-Python library, kinterbasdb.

### **A.3. Selected Source Codes from Muscorian**

#### **Path Name: muscorian/BioDatabases/musclopedia.sql**

```

/* =====
Initial database file is set to 4GB (4KB x 1000000).

Prefix / Suffix defined:
    ec_: enzyme commission
    go_: gene ontology
    pg_: protein and gene names
    pmc_: pub med corpus
    ml_: montylingua

(reference to page 23 of book 217, dated 15th Nov 2004)
(reference to page 18 of e-notebook, annotated on 17th Dec 2004)

Copyright 2004 Maurice Ling. All rights reserved.
=====*/

/*=====*/
/* environment setting */
/*=====*/
set sql dialect 3;

/*=====*/
/* database creation */
/*=====*/
create database 'musclopedia.fdb'
    user 'mouse' password 'mouse'
    page_size 4096
    length 1000000;

CREATE DOMAIN "ID" AS NUMERIC(18, 0);

/*=====*/
/* table definitions (metadata) ----- */
/*=====*/
create table m$revision (
    m$date date,
    m$key varchar(50) not null,
    m$value varchar(8000),
    ID ID not null);

create table m$system (
    m$key varchar(50) not null primary key,
    m$value varchar(8000),
    ID ID not null);

create table m$information (
    m$key varchar(50) not null,
    m$value varchar(8000),
    ID ID not null);

/*=====*/
/* table definitions (pubmed corpus) ----- */
/*=====*/
create table pmc_abstract(
    pmid numeric(10,0) not null primary key,
    text varchar(18000),
    ablen numeric(5,0) default 0 not null,
    ID ID not null);

create table pmc_article_info(

```

```

        pmid numeric(10,0) not null primary key,
        title char(150) default 'not_set',
        issn char(9) default '0000-0000',
        issue char(5),
        volume char(5),
        pagination char(12),
        doi char(50),
        ID ID not null);

create table pmc_author(
    pmid numeric(10,0) not null,
    lastname char(30),
    initial char(30),
    forename char(30),
    ID ID not null);

create table pmc_index_list(
    termid numeric(5,0) not null primary key,
    term char(30) not null,
    indextype char(30) default 'all_fields',
    ID ID not null);

create table pmc_index_term_map(
    pmid numeric(10,0) not null,
    termid numeric(5,0) not null,
    ID ID not null);

create table abstract_index(
    pmid numeric(10,0) not null,
    term varchar(250) not null,
    offset numeric(5,0) not null,
    primary key (pmid, offset),
    ID ID not null);

create table text_replace_abstract(
    pmid numeric(10,0) not null,
    text varchar(18000),
    ID ID not null);

create table text_replace_abstract_upro(
    pmid numeric(10,0) not null,
    text varchar(18000));

/*=====*/
/* Tables creation - montylingua tables */
/*=====*/

/* SVO output of MontyLingua */
create table ml_svo (
    pmid numeric(10,0) not null,
    verb char(30) not null,
    subject char(50) not null,
    object char(50) not null,
    ID ID not null,
    primary key (pmid, subject, verb, object));

create table ppi_pmid (
    pmid numeric(10,0) not null,
    verb char(30) not null,
    subject char(50) not null,
    object char(50) not null,
    version char(10) not null,
    ID ID not null,
    primary key (pmid, subject, verb, object, version));

create table ppi_distinct (
    subject char(50) not null,
    object char(50) not null);

/* subset of ml_svo, protein-protein activation, verb='activate' */
create table activate_pmid(
    pmid numeric(10,0) not null,
    subject char(50) not null,
    object char(50) not null,
    version char(10) default '0',
    ID ID not null,
    constraint activate_unique unique (pmid, subject, object, version));

```



```

/* subset of ml_svo, protein-protein binding, verb='bind' */
create table bind_pmid(
    pmid numeric(10,0) not null,
    subject char(50) not null,
    object char(50) not null,
    version char(10) default '0',
    ID ID not null,
    constraint bind_unique unique (pmid, subject, object, version));

/* subset of ml_svo, protein-protein degradation, verb='degrade' */
create table degrade_pmid(
    pmid numeric(10,0) not null,
    subject char(50) not null,
    object char(50) not null,
    version char(10) default '0',
    ID ID not null,
    constraint degrade_unique unique (pmid, subject, object, version));

/* subset of ml_svo, protein-protein enhancement, verb='enhance' */
create table enhance_pmid(
    pmid numeric(10,0) not null,
    subject char(50) not null,
    object char(50) not null,
    version char(10) default '0',
    ID ID not null,
    constraint enhance_unique unique (pmid, subject, object, version));

/* subset of ml_svo, protein-protein inhibition, verb='inhibit' */
create table inhibit_pmid(
    pmid numeric(10,0) not null,
    subject char(50) not null,
    object char(50) not null,
    version char(10) default '0',
    ID ID not null,
    constraint inhibit_unique unique (pmid, subject, object, version));

create table bind_distinct (
    subject char(50) not null,
    object char(50) not null);

create table activate_distinct (
    subject char(50) not null,
    object char(50) not null);

/* subset of ml_svo, protein-disease relation */
create table protein_disease(
    pmid numeric(10,0) not null,
    verb char(30) not null,
    disease char(50) not null,
    protein char(50) not null,
    ID ID not null);

/*=====*/
/* Table creation - protein/gene name hunting (a posteriori name co-occurrence
statistics)*/
/*=====*/
create table name_hunt_waiting (
    pmid numeric(10,0) not null);

/* list of predicted protein and gene names in abstract */
create table pg_name (
    pmid numeric(10,0) not null,
    name varchar(125) not null,
    start numeric(3,0) not null,
    primary key (pmid, name, start));

/* set of unique protein and gene names, from pg_name table */
create table pg_distinct(
    name varchar(125) not null,
    primary key (name));

/* list of pmds having pairwise protein or gene names in abstract */
create table pg_comb(
    pmid numeric(10,0) not null,
    name1 varchar(125) not null,
    name2 varchar(125) not null);

```

```

/* mapping of predicted protein and gene names with GenBank ID */
create table pg_id(
    name varchar(125) not null,
    gi numeric(10,0) not null,
    primary key(name, gi));

/*=====*/
/* Table creation - jobs tables */
/*=====*/
create table job_master(
    jobid numeric (5,0) not null primary key,
    description varchar(1024) not null,
    submitter char(50) default 'system' not null,
    ID ID not null);

create table job_clone_master(
    jclone char(128) not null primary key,
    ID ID not null);

create table job_clone_map(
    jclone char(128) not null,
    other_id char(15) not null,
    idtype char(10) not null,
    ID ID not null,
    primary key(jclone, other_id, idtype));

create table job_clone(
    jobid numeric (5,0) not null,
    jclone char(128) not null,
    ID ID not null,
    primary key(jobid, jclone));

create table job_interaction(
    jobid numeric (5,0) not null,
    interaction char(30) default 'activate' not null,
    ID ID not null,
    primary key(jobid, interaction));

/*=====*/
/* Table creation - microarray results */
/*=====*/
create table microarray_main(
    microarrayid numeric (5,0) not null primary key,
    description varchar(1024) not null,
    submitter char(50) default 'system' not null,
    ID ID not null);

create table microarray_experiment(
    experiment_id numeric (5,0) not null,
    microarrayid numeric (5,0) not null,
    ID ID not null,
    primary key(experiment_id, microarrayid));

create table microarray_clone_master(
    clone_id char(50) not null primary key,
    description varchar(128) default 'unknown' not null,
    ID ID not null);

create table microarray_clone_map(
    clone_id char(50) not null,
    other_id char(15) not null,
    idtype char(10) not null,
    ID ID not null,
    primary key(clone_id, other_id, idtype));

create table microarray_data_abs(
    microarrayid numeric (5,0) not null,
    clone_id char(50) not null,
    intensity numeric(6,3) default '000000.000' not null,
    ID ID not null,
    primary key(microarrayid, clone_id));

create table microarray_stats(
    dtype char(30),
    dvalue char(30),
    ID ID not null);

```

```

create table mdata_correlation(
    clone_id1 char(50) not null,
    clone_id2 char(50) not null,
    correlation numeric(3, 3) default '000.000' not null);

/*=====*/
/* Table creation - microarray/job match */
/*=====*/
create table jclone_mclone(
    jclone char(128) not null,
    clone_id char(50) not null,
    ID ID not null,
    primary key(jclone, clone_id));

/*=====*/
/* Table creation - clone description soundex */
/*=====*/
create table clone_soundex(
    cloneid char(50) not null, /*microarray_clone_map.clone_id*/
    otherid char(15) not null,
    idtype char(10) not null,
    desc_sound char(20) default '00000' not null,
    abb_sound char(5) default '00000' not null,
    ID ID not null);

/*=====*/
/* Table creation - a priori name co-occurrence */
/*=====*/
create table jclone_occurrence(
    jclone char(128) not null primary key,
    occurrence numeric(7,0),
    ID ID not null);

create table jclone_occurrence_pmid(
    pmid numeric(10,0) not null,
    clone char(128) not null,
    ID ID not null);

create table jclone_occur_stat(
    clone1 char(128) not null,
    clone2 char(128) not null,
    randomoccur numeric(6,6) default '000000.000000' not null,
    poisson95 numeric(6,0) default '000000' not null,
    poisson99 numeric(6,0) default '000000' not null,
    ID ID not null);

create table jclone_occur_pair(
    clone1 char(128) not null,
    clone2 char(128) not null,
    paircount numeric(10,0) not null,
    ID ID not null);

create table jclone_occur_single(
    clone char(128) not null,
    paircount numeric(10,0) not null,
    ID ID not null);

/*=====*/
/* Table creation - others */
/*=====*/
create table pg1_verb(
    subject char(50) not null,
    verb char(30) not null,
    object char(50));

create table bindcount(
    clone1 char(100),
    clone2 char(100),
    occur char(100));

create table activatecount(
    clone1 char(100),
    clone2 char(100),
    occur char(100));

```

```

create table degradecount(
  clone1 char(100),
  clone2 char(100),
  occur char(100));

create table enhancecount(
  clone1 char(100),
  clone2 char(100),
  occur char(100));

create table inhibitcount(
  clone1 char(100),
  clone2 char(100),
  occur char(100));

create table acc_ug_ll(
  accession char(12),
  unigene char(12),
  locuslink char(10));

create table affy_annot(
  probeid char(35),
  unigenecluster char(12),
  unigene char(12),
  symbol char(40),
  entrez char(30),
  description char(128),
  ensembl char(30));

create table cw_inv_coexp(
  probe1 char(35),
  probe2 char(35),
  correlation numeric(3,3) default '000.000' not null);

create table km_mouse_vitro(
  probeid char(35),
  sFP12 numeric(6,3) default '000000.000' not null,
  sIF12 numeric(6,3) default '000000.000' not null,
  sIFP12 numeric(6,3) default '000000.000' not null,
  s8 numeric(6,3) default '000000.000' not null,
  sIFPG10 numeric(6,3) default '000000.000' not null,
  sIFPG12 numeric(6,3) default '000000.000' not null,
  s12 numeric(6,3) default '000000.000' not null,
  sIP10 numeric(6,3) default '000000.000' not null,
  sFP10 numeric(6,3) default '000000.000' not null,
  sIF10 numeric(6,3) default '000000.000' not null,
  sIFP10 numeric(6,3) default '000000.000' not null,
  sIFG10 numeric(6,3) default '000000.000' not null,
  sIFG12 numeric(6,3) default '000000.000' not null,
  s10 numeric(6,3) default '000000.000' not null,
  sIP12 numeric(6,3) default '000000.000' not null);

create table ts_array_data (
  unigene char(12),
  V10_A numeric(6,3) default '000000.000' not null,
  V10_B numeric(6,3) default '000000.000' not null,
  V10_C numeric(6,3) default '000000.000' not null,
  V12_A numeric(6,3) default '000000.000' not null,
  V12_B numeric(6,3) default '000000.000' not null,
  V12_C numeric(6,3) default '000000.000' not null,
  P1_A numeric(6,3) default '000000.000' not null,
  P1_B numeric(6,3) default '000000.000' not null,
  P1_C numeric(6,3) default '000000.000' not null,
  P2_A numeric(6,3) default '000000.000' not null,
  P2_B numeric(6,3) default '000000.000' not null,
  P2_C numeric(6,3) default '000000.000' not null,
  P3_A numeric(6,3) default '000000.000' not null,
  P3_B numeric(6,3) default '000000.000' not null,
  P3_C numeric(6,3) default '000000.000' not null,
  P8_5_A numeric(6,3) default '000000.000' not null,
  P8_5_B numeric(6,3) default '000000.000' not null,
  P8_5_C numeric(6,3) default '000000.000' not null,
  P12_5_A numeric(6,3) default '000000.000' not null,
  P12_5_B numeric(6,3) default '000000.000' not null,
  P12_5_C numeric(6,3) default '000000.000' not null,
  P14_5_A numeric(6,3) default '000000.000' not null,
  P14_5_B numeric(6,3) default '000000.000' not null,

```

```

P14_5_C numeric(6,3) default '000000.000' not null,
P17_5_A numeric(6,3) default '000000.000' not null,
P17_5_B numeric(6,3) default '000000.000' not null,
P17_5_C numeric(6,3) default '000000.000' not null,
Lac1_A numeric(6,3) default '000000.000' not null,
Lac1_B numeric(6,3) default '000000.000' not null,
Lac1_C numeric(6,3) default '000000.000' not null,
Lac3_A numeric(6,3) default '000000.000' not null,
Lac3_B numeric(6,3) default '000000.000' not null,
Lac3_C numeric(6,3) default '000000.000' not null,
Lac7_A numeric(6,3) default '000000.000' not null,
Lac7_B numeric(6,3) default '000000.000' not null,
Lac7_C numeric(6,3) default '000000.000' not null,
Inv1_A numeric(6,3) default '000000.000' not null,
Inv1_B numeric(6,3) default '000000.000' not null,
Inv1_C numeric(6,3) default '000000.000' not null,
Inv2_A numeric(6,3) default '000000.000' not null,
Inv2_B numeric(6,3) default '000000.000' not null,
Inv2_C numeric(6,3) default '000000.000' not null,
Inv3_A numeric(6,3) default '000000.000' not null,
Inv3_B numeric(6,3) default '000000.000' not null,
Inv3_C numeric(6,3) default '000000.000' not null,
Inv4_A numeric(6,3) default '000000.000' not null,
Inv4_B numeric(6,3) default '000000.000' not null,
Inv4_C numeric(6,3) default '000000.000' not null,
Inv20_A numeric(6,3) default '000000.000' not null,
Inv20_B numeric(6,3) default '000000.000' not null,
Inv20_C numeric(6,3) default '000000.000' not null);

create table pn_array_data(
  probeid char(35),
  DF40_1 numeric(6,3) default '000000.000' not null,
  DF40_2 numeric(6,3) default '000000.000' not null,
  DF40_3 numeric(6,3) default '000000.000' not null,
  DF8_1 numeric(6,3) default '000000.000' not null,
  DF8_2 numeric(6,3) default '000000.000' not null,
  DF8_3 numeric(6,3) default '000000.000' not null,
  DF16_1 numeric(6,3) default '000000.000' not null,
  DF16_2 numeric(6,3) default '000000.000' not null,
  DF16_3 numeric(6,3) default '000000.000' not null,
  NFD2lact1 numeric(6,3) default '000000.000' not null,
  NFD2lact2 numeric(6,3) default '000000.000' not null,
  NFD2lact3 numeric(6,3) default '000000.000' not null,
  NFD2lact4 numeric(6,3) default '000000.000' not null,
  NFD9lact1 numeric(6,3) default '000000.000' not null,
  NFD9lact2 numeric(6,3) default '000000.000' not null,
  NFD9lact3 numeric(6,3) default '000000.000' not null,
  NFD9lact4 numeric(6,3) default '000000.000' not null,
  NFD12Preg1 numeric(6,3) default '000000.000' not null,
  NFD12Preg2 numeric(6,3) default '000000.000' not null,
  NFD12Preg3 numeric(6,3) default '000000.000' not null,
  NFD12Preg4 numeric(6,3) default '000000.000' not null,
  NFD1Lact1 numeric(6,3) default '000000.000' not null,
  NFD1Lact2 numeric(6,3) default '000000.000' not null,
  NFD1Lact3 numeric(6,3) default '000000.000' not null,
  NFD1Lact4 numeric(6,3) default '000000.000' not null,
  NFD17Preg1 numeric(6,3) default '000000.000' not null,
  NFD17Preg2 numeric(6,3) default '000000.000' not null,
  NFD17Preg3 numeric(6,3) default '000000.000' not null,
  NFD17Preg4 numeric(6,3) default '000000.000' not null);

create table ma_statistics_bt(
  unigene char(12),
  mean numeric(6,3),
  sd numeric(6,3))

create table km_insulin1(
  unigeneclass char(12),
  unigene char(12),
  ratio numeric(6,3) default '000000.000' not null);

create table km_glucocorticoid1(
  unigeneclass char(12),
  unigene char(12),
  ratio numeric(6,3) default '000000.000' not null);

create table km_prolactin1(

```

```

        unigenecluster char(12),
        unigene char(12),
        ratio numeric(6,3) default '000000.000' not null);

create table ts_coexp_as_unigene(
    unigene1 char(12),
    unigene2 char(12),
    correlation numeric(3,3) default '000.000' not null);

create table pn_coexp(
    clone1 char(35),
    clone2 char(35),
    correlation numeric(3,3) default '000.000' not null);

create table pnts_intersect(
    unigene1 char(12),
    unigene2 char(12));

create table pntscw_intersect(
    unigene1 char(12),
    unigene2 char(12));

create table pntscwlc_intersect(
    unigene1 char(12),
    unigene2 char(12));

create table pubgene_cache(
    unigene1 char(12),
    unigene2 char(12),
    paircount numeric(10,0),
    clone1 char(128),
    clone2 char(128));

create table all_coexp(
    unigene1 char(12),
    unigene2 char(12));

create table lactogenesis_up (
    unigene char(12));

create table lactogenesis_down (
    unigene char(12));

create table IFPdiff(
    diff char(5),
    unigene char(12));

create table galactopoesis_up (
    unigene char(12));

create table galactopoesis_down (
    unigene char(12));

/*=====*/
/* views definitions ----- */
/*=====*/
create view clone_view (cloneid, otherid, idtype, description, abb_desc,
unigene, locuslink) as
    select mmap.clone_id, mmap.other_id, mmap.idtype, mmap.description,
           jmap.jclone, aul.unigene, aul.locuslink
    from microarray_clone_map mmap left outer join
         microarray_clone_master mmapster
         on mmap.clone_id=mmapster.clone_id
    left outer join job_clone_map jmap
         on mmap.other_id=jmap.other_id
    left outer join acc_ug_ll aul
         on mmap.other_id=aul.accession;

create view bind_t (s_unigene, subject, object, pmid) as
    select c.unigene as s_unigene, b.subject, b.object, b.pmid
    from bind_pmid b join clone_view c on b.subject=c.abb_desc
    where b.version = 80;

create view bind_as_unigene (s_unigene, o_unigene, subject, object, pmid) as
    select b.s_unigene, c.unigene as o_unigene, b.subject, b.object, b.pmid
    from bind_t b join clone_view c on b.object=c.abb_desc;

```

```

create view activate_t (s_unigene, subject, object, pmid) as
select c.unigene as s_unigene, a.subject, a.object, a.pmid
from activate_pmid a join clone_view c on a.subject=c.abb_desc
where a.version = 80;

create view activate_as_unigene (s_unigene, o_unigene, subject, object, pmid)
as
select a.s_unigene, c.unigene as o_unigene, a.subject, a.object, a.pmid
from activate_t a join clone_view c on a.object=c.abb_desc;

create view pn_coexp_t (unigenel, clone2, correlation, clone1) as
select a.unigenel as unigenel, p.clone2, p.correlation, p.clone1
from pn_coexp p join affy_annot a on p.clone1=a.probeid;

create view pn_coexp_as_unigene (unigenel, unigene2, correlation, clone1,
clone2) as
select p.unigenel, a.unigene as unigene2, p.correlation, p.clone1,
p.clone2
from pn_coexp_t p join affy_annot a on p.clone2=a.probeid;

create view cw_coexp_t (unigenel, clone2, correlation, clone1) as
select a.unigene as unigenel, c.probe2 as clone2, c.correlation,
c.probel as clone1
from cw_inv_coexp c join affy_annot a on c.probel=a.probeid;

create view cw_coexp_as_unigene (unigenel, unigene2, correlation, clone1,
clone2) as
select c.unigenel, a.unigene as unigene2, c.correlation, c.clone1,
c.clone2
from cw_coexp_t c join affy_annot a on c.clone2=a.probeid;

create view lc_coexp_t (unigenel, clone2, correlation, clone1) as
select c.unigene as unigenel, m.clone_id2 as clone2, m.correlation,
m.clone_id1 as clone1
from mdata_correlation m join clone_view c on m.clone_id1=c.cloneid;

create view lc_coexp_as_unigene (unigenel, unigene2, correlation, clone1,
clone2) as
select m.unigenel, c.unigene as unigene2, m.correlation, m.clone1,
m.clone2
from lc_coexp_t m join clone_view c on m.clone2=c.cloneid;

create view pubgene_t (unigenel, clone2, paircount, clone1) as
select c.unigene as unigenel, p.clone2, p.paircount, p.clone1
from jclone_occur_pair p join clone_view c on p.clone1=c.abb_desc
where c.unigene is not null and p.paircount > 0;

create view pubgene_as_unigene (unigenel, unigene2, paircount, clone1, clone2)
as
select p.unigenel, c.unigene as unigene2, p.paircount, p.clone1,
p.clone2
from pubgene_t p join clone_view c on p.clone2=c.abb_desc
where c.unigene is not null;

create view km_mouse_vitro_id (probeid, unigenecluster, unigene, entrez,
accession, locuslink, symbol, SFP12, SIF12, SIFP12, S8, SIFPG10, SIFPG12, S12,
SIP10, SFP10, SIF10, SIFP10, SIFG10, SIFG12, S10, SIP12) as
select a.probeid, a.unigenecluster, aul.unigene, a.entrez,
aul.accession, aul.locuslink, a.symbol, k.SFP12, k.SIF12,
k.SIFP12, k.S8, k.SIFPG10, k.SIFPG12, k.S12, k.SIP10, k.SFP10,
k.SIF10, k.SIFP10, k.SIFG10, k.SIFG12, k.S10, k.SIP12
from affy_annot a inner join km_mouse_vitro k on a.probeid=k.probeid
inner join acc_ug_ll aul on a.unigene=aul.unigene;

/*=====*/
/* index definitions ----- */
/*=====*/
create index m$date_index on m$revision (m$date);

create descending index pmc_article_pmid on pmc_article_info (pmid);
create descending index pmc_abstract_pmid on pmc_abstract (pmid);

create descending index jclone_occur_pmid_clone on jclone_occurrence_pmid
(clone);
create index jclone_occur_pmid_clonepmid on jclone_occurrence_pmid (clone,
pmid);

```

```

create index clone_soundex_cloneid on clone_soundex (cloneid);
create index clone_soundex_oid on clone_soundex (otherid);
create index clone_soundex_desc on clone_soundex (desc_sound);
create index clone_soundex_abb on clone_soundex (abb_sound);

create index km_mouse_vitro_id on km_mouse_vitro (probeid);
create index ACC_UG_LL_ACCESSION ON ACC_UG_LL(ACCESSION);
create index ACC_UG_LL_UNIGENE ON ACC_UG_LL(UNIGENE);
create index AFFY_ANNOT_PROBEID ON AFFY_ANNOT(PROBEID);
create index affy_annot_unigene on affy_annot (unigene);
create index JCLONE_OCCUR_PAIR_C1 ON JCLONE_OCCUR_PAIR(CLONE1);
create index JCLONE_OCCUR_PAIR_C2 ON JCLONE_OCCUR_PAIR(CLONE2);
create index JCLONE_OCCUR_STAT_C1 ON JCLONE_OCCUR_STAT(CLONE1);
create index JCLONE_OCCUR_STAT_C2 ON JCLONE_OCCUR_STAT(CLONE2);
create index MICROARRAY_CLONE_MAP_CLONEID ON MICROARRAY_CLONE_MAP(CLONE_ID);
create index MICROARRAY_CLONE_MAP_OTHERID ON MICROARRAY_CLONE_MAP(OTHER_ID);
create index job_clone_map_otherid on job_clone_map (other_id);
create index mdata_correlaction_clone on mdata_correlation (clone_id1,
clone_id2);
create index cw_inv_coexp_clone on cw_inv_coexp (probe1, probe2);
create index ts_unigenel on ts_coexp_as_unigene (unigenel);
create index ts_unigene2 on ts_coexp_as_unigene (unigene2);
create index pn_coexp_clone1 on pn_coexp (clone1);
create index pn_coexp_clone2 on pn_coexp (clone2);
create index pnts_1 on pnts_intersect (unigenel);
create index pnts_2 on pnts_intersect (unigene2);
create index pntscw_1 on pntscw_intersect (unigenel);
create index pntscw_2 on pntscw_intersect (unigene2);
create index pntscwlc_1 on pntscwlc_intersect (unigenel);
create index pntscwlc_2 on pntscwlc_intersect (unigene2);
create index pubgene_cache_ug1 on pubgene_cache (unigenel);
create index pubgene_cache_ug2 on pubgene_cache (unigene2);
create index pubgene_cache_c1 on pubgene_cache (clone1);
create index pubgene_cache_c2 on pubgene_cache (clone2);
create index ts_array_unigene on ts_array_data (unigene);
create index pn_array_probeid on pn_array_data (probeid);

/*=====*/
/* generator definitions ----- */
/*=====*/
create generator pmc_termid_gen;
set generator pmc_termid_gen to 10000;
create generator job_id_gen;
set generator job_id_gen to 10000;
create generator microarray_id_gen;
set generator microarray_id_gen to 10000;
create generator mexperiment_id_gen;
set generator mexperiment_id_gen to 10000;
CREATE GENERATOR "ID";
CREATE GENERATOR MRChangeID;

/*=====*/
/* trigger definitions ----- */
/*=====*/
set term !! ; ;

create trigger pmc_index_list_id for pmc_index_list
before insert position 0
as begin
    if (NOT (user='FIBRE')) then begin
        new.id = gen_id(ID, 1);
        new.termid = gen_id(pmc_termid_gen, 1);
    end
end !!

create trigger job_master_id for job_master
before insert position 0
as begin
    if (NOT (user='FIBRE')) then begin
        new.id = gen_id(ID, 1);
        if (new.jobid is null) then
            begin
                new.jobid = gen_id(job_id_gen, 1);
            end
        end
    end
end

```



```

end !!

create trigger microarray_main_id for microarray_main
before insert position 0
as begin
    if (NOT (user='FIBRE')) then begin
        new.id = gen_id(ID, 1);
        if (new.microarrayid is null) then
            begin
                new.microarrayid = gen_id(microarray_id_gen, 1);
            end
        end
    end
end !!

create trigger microarray_experiment_id for microarray_experiment
before insert position 0
as begin
    if (NOT (user='FIBRE')) then begin
        new.id = gen_id(ID, 1);
        if (new.experiment_id is null) then
            begin
                new.experiment_id = gen_id(mexperiment_id_gen, 1);
            end
        end
    end
end !!

set term ; !! !!

/* metadata inserts
The following statements MUST be executed to enter the values into the
database. Any attempt to avoid the execution of these statements; or amend or
delete these statements, before or after execution by a SQL client; or amend
or delete the corresponding entries in the database after the execution of
these statements; or any malicious efforts, intended or unintended, for the
purpose of, or resulting in discreditation of work is illegal and accounts for
explicit violation of any forms of agreements, explicit or implied, without
prior written permission from Maurice Ling. In event whereby execution of
these statement(s) fail(s), it will be the onus of the user to enter these
entries into the database or to perform any necessary act(s) to result in the
same intended effect as executing these statements. Do not use this database
or subset of this database if any of the above terms is disagreeable or if you
do not have the right to agree to these terms.
-----*/
insert into m$information (m$key, m$value) values ('creator', 'Maurice H.T.
Ling <mauriceling@acm.org>');
insert into m$information (m$key, m$value) values ('copyright', 'Copyright
2004 Maurice H.T. Ling. All rights reserved.');
```

insert into m\$information (m\$key, m\$value) values ('project support', 'This database is part of the Ph.D. work of Maurice H.T. Ling at the Department of Zoology, The University of Melbourne, Australia, under the financial support from the Cooperative Research Centre for Innovative Dairy Products, Australia. This Ph.D. work is currently under the supervision of Associate Professor Kevin R. Nicholas, Principle Research Fellow at the Department of Zoology, The University of Melbourne, and Dr. Christophe Leferve of Victorian Bioinformatics Consortium, Australia.');

insert into m\$information (m\$key, m\$value) values ('project support', 'Dr. Andrew Lonie of Department of Information Systems, The University of Melbourne, has joined as a co-supervisor of Maurice Ling's Ph.D. with respect from 22nd June 2005.');

insert into m\$information (m\$key, m\$value) values ('project support', 'Associate Professor Lin Feng of Bioinformatics Research Centre, School of Computer Engineering, Nanyang Technological University, Singapore, is co-supervising Maurice Ling's PhD from 7th October 2005 to 23rd April 2006. During this period, Maurice is physically attached to Bioinformatics Research Centre (BIRC), NTU, Singapore.');

insert into m\$information (m\$key, m\$value) values ('project support', 'Dr. Andrew Lonie of Department of Information Systems, The University of Melbourne, has resigned as a co-supervisor of Maurice Ling's Ph.D. with respect from 1st March 2007.');

```

/* end of metadata inserts ----- */

/* alter database statements ----- */
/* alter database add file 'muscopedia.fb2' starting at page 1000001 length
2000000;
alter database add file 'muscopedia.fb3' starting at page 2000001 length

```

```

1000000;
alter database add file 'muscopedia.fb4' starting at page 3000001 length
1000000;
alter database add file 'muscopedia.fb5' starting at page 4000001 length
1000000;
alter database add file 'muscopedia.fb6' starting at page 5000001 length
1000000;
alter database add file 'muscopedia.fb7' starting at page 6000001 length
1000000;
alter database add file 'muscopedia.fb8' starting at page 7000001 length
1000000;
alter database add file 'muscopedia.fb9' starting at page 8000001 length
1000000; */
commit;

```

## Path Name: muscorian/muscorian.py

"""

File name: muscorian.py

This is the command line interface for Muscorian.

Author: Maurice Ling, The University of Melbourne

Date Created: 7th April 2005

"""

```

from muscoInform import *
import montylingua
from abcrawl import PubMedGrabber
import UCFunctions
import traceback
import string
import sys
import math
import os
import marshal
import random
import gc
from time import sleep

RRFile = ['RRULE_aa', 'RRULE_ab', 'RRULE_ac', 'RRULE_ad', 'RRULE_ae']

class MuscorianSystem:
    environment = {}
    result = []
    pubmed = None
    montylingua = None
    muscopedia = None

    def environment_initialize(self):
        """
        Method to initialize the system when it starts up, before accepting
        any commands from the user. Start up tasks:
        1. to load the a set of environment variables from a file,
           mosirium_environment.ini, which is supposedly located in the same
           directory as this application. If the file is not present, it
           populates the system with a default set of environment
           variables."""
        # task 1
        if (os.path.isfile(os.path.join(os.getcwd(),
                                         "muscorian_environment.ini"))):
            f = open(os.path.join(os.getcwd(), "muscorian_environment.ini"),
                    'r')
            self.environment = marshal.load(f)
            f.close()
            gc.enable()
        else:
            self.environment["inidir"] = os.getcwd()
            self.environment["cwd"] = os.getcwd()
            self.environment["diary"] = "on"
            self.environment["diaryfile"] = os.path.join(os.getcwd(),
                                                         "diary.txt")
            gc.enable()

    def PreRun(self, command):

```

```

"""
Method to perform necessary operations before a command is executed.
Current operations performed:
    1. append the command into the diary file, if set on."""
if (self.environment["diary"] == "on"):
    try:
        filename = self.environment["diaryfile"]
        f = open(filename, "a+")
        f.write(str(command) + os.linesep)
        f.close()
    except: print "Unable to open diary file : " + filename

def PostRun(self, result):
    """
    Method to perform necessary operations after a command is executed.
    Current operations performed:
        1. append the results into the diary file, if set on."""
    if (self.environment["diary"] == "on"):
        try:
            filename = self.environment["diaryfile"]
            f = open(filename, "a+")
            for line in result:
                f.write(str(line) + os.linesep)
            f.close()
        except: print "Unable to open diary file : " + filename

def UnknownCommandError(self):
    """
    A generic method that is invoked when a command parsing event fails
    (command is not recognized). It displays a set of default messages when
    invoked."""
    self.result.append("Command : " + str(command[0]) + ", is
        unrecognized.")
    self.result.append("Please take absolute note that all commands must
        be in lowercase.")
    self.result.append("Please type 'help' for the list of commands.")

# -----
# Command Processors
# -----

def commandAbstractSort(self):
    """Invoked by 'abstractsort' command. It calls commandASactivate0(),
    commandASbind0(), commandASdegrade0(), commandASenhance0(), and
    commandASinhibit0() methods.

    Usage: abstractsort"""
    self.commandASactivate0()
    self.commandASbind0()
    self.commandASdegrade0()
    self.commandASenhance0()
    self.commandASinhibit0()

def commandAbstract80(self):
    """Invoked on 'abstract80' command. It executes to the equivalent
    effect of running commandASactivate80(), commandASbind80(),
    commandASdegrade80(), commandASenhance80(), and commandASinhibit80()
    methods.

    Usage: abstract80
    """
    self.muscopedia.dbcursor.execute('select jclone from
        job_clone_master')
    clone = [clone[0].strip() for clone in
        self.muscopedia.dbcursor.fetchall()]
    count = 0
    for table in ['activate_pmid', 'bind_pmid', 'enhance_pmid',
        'degrade_pmid', 'inhibit_pmid']:
        for subj in clone:
            for obj in clone:
                count = count + 1
                sqlstmt = "insert into %s (pmid, subject, object, \
                    version) \
                    select pmid, '%s', '%s', '80' from %s where \
                    version = '0' and subject containing '%s' \
                    and object containing '%s'" % (table, subj, \
                    obj, table, subj, obj)

```

```

        try:
            self.muscopedia.dbcursor.execute(sqlstmt)
        except: pass
        if (count % 1000) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' interacting pairs
                        processed for ' + table])
            print str(count) + ' interacting pairs processed for '
              + table
            self.PostRun([str(table) + ' table processed'])
            print str(table) + ' table processed'

def commandActivateEDA(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from \
        activate_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    self.muscopedia.dbcursor.execute("delete from activatecount")
    self.muscopedia.dbconnect.commit()
    clone = amap._clean_combination()
    pfile = open('p'+command[1], 'w')
    cfile = open('c'+command[1], 'w')
    for k in clone.keys():
        temp = k.split('--')
        pfile.write(temp[0] + '\t(pp)\t' + temp[1] + ' = activate' +
            os.linesep)
        cfile.write(temp[0] + '\t(pp)\t' + temp[1] + ' = ' + str(clone[k])
            + os.linesep)
        self.muscopedia.dbcursor.execute("insert into activatecount \
            (clone1, clone2, occur) values (?, ?, ?)",
            (temp[0], temp[1], str(clone[k])))
    self.muscopedia.dbconnect.commit()
    pfile.close()

def commandActivateSIF(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from \
        activate_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    clone = amap._clean_combination()
    filename = open(command[1], 'w')
    for k in clone.keys():
        temp = k.split('--')
        filename.write(temp[0] + '\tpp\t' + temp[1])
    filename.close()

def commandASactivate0(self):
    """
    Invoked on 'asactivate0' command. It loads the assertions from ml_svo
    table where verb is 'activate' into activate_pmid table as version '0'
    and changed the 'subject' and 'object' into upper case.

    Usage: asactivate0"""
    self.muscopedia.dbcursor.execute("delete from activate_pmid where \
        version = '0'")
    sqlstmt = "insert into activate_pmid (pmid, subject, object, version) \
        select distinct pmid, upper(subject), upper(object), '0' \
        from ml_svo where verb = 'activate'"
    self.muscopedia.dbcursor.execute(sqlstmt)
    self.muscopedia.dbconnect.commit()

def commandASbind0(self):
    """
    Invoked on 'asbind0' command. It loads the assertions from ml_svo
    table where verb is 'bind' into bind_pmid table as version '0' and
    changed the 'subject' and 'object' into upper case.

    Usage: asbind0"""
    self.muscopedia.dbcursor.execute("delete from bind_pmid where \
        version = '0'")
    sqlstmt = "insert into bind_pmid (pmid, subject, object, version) \
        select distinct pmid, upper(subject), upper(object), '0' \
        from ml_svo where verb = 'bind'"
    self.muscopedia.dbcursor.execute(sqlstmt)
    self.muscopedia.dbconnect.commit()

```

```

def commandASdegrade0(self):
    """
    Invoked on 'asdegrade0' command. It loads the assertions from ml_svo
    table where verb is 'degrade' into degrade_pmid table as version '0'
    and changed the 'subject' and 'object' into upper case.

    Usage: asdegrade0"""
    self.muscopedia.dbcursor.execute("delete from degrade_pmid where \
        version = '0'")
    sqlstmt = "insert into degrade_pmid (pmid, subject, object, version) \
        select distinct pmid, upper(subject), upper(object), '0' \
        from ml_svo where verb = 'degrade'"
    self.muscopedia.dbcursor.execute(sqlstmt)
    self.muscopedia.dbconnect.commit()

def commandASenhance0(self):
    """
    Invoked on 'asenhance0' command. It loads the assertions from ml_svo
    table where verb is 'enhance' into enhance_pmid table as version '0' and
    changed the 'subject' and 'object' into upper case.

    Usage: asenhance0"""
    self.muscopedia.dbcursor.execute("delete from enhance_pmid where \
        version = '0'")
    sqlstmt = "insert into enhance_pmid (pmid, subject, object, version) \
        select distinct pmid, upper(subject), upper(object), '0' \
        from ml_svo where verb = 'enhance'"
    self.muscopedia.dbcursor.execute(sqlstmt)
    self.muscopedia.dbconnect.commit()

def commandASinhibit0(self):
    """
    Invoked on 'asinhibit0' command. It loads the assertions from ml_svo
    table where verb is 'inhibit' into inhibit_pmid table as version '0' and
    changed the 'subject' and 'object' into upper case.

    Usage: asinhibit0"""
    self.muscopedia.dbcursor.execute("delete from inhibit_pmid where \
        version = '0'")
    sqlstmt = "insert into inhibit_pmid (pmid, subject, object, version) \
        select distinct pmid, upper(subject), upper(object), '0' \
        from ml_svo where verb = 'inhibit'"
    self.muscopedia.dbcursor.execute(sqlstmt)
    self.muscopedia.dbconnect.commit()

def commandASactivate80(self):
    """
    Invoked on 'asactivate80' command. It processes version 0 of
    activate_pmid table to look for SVOs containing entities in
    job_clone_master table in both subject and object attributes. For SVOs
    whose subject and object attributes contain entities in job_clone_master
    table (abbreviated clone names), the attributes were replaced with the
    abbreviated clone name and tagged as version 80.

    Usage: asactivate80
    """
    self.muscopedia.dbcursor.execute('select jclone from \
        job_clone_master')
    clone = [clone[0].strip() for clone in
        self.muscopedia.dbcursor.fetchall()]
    count = 0
    for table in ['activate_pmid']:
        for subj in clone:
            for obj in clone:
                count = count + 1
                sqlstmt = "insert into %s (pmid, subject, object, \
                    version) select pmid, '%s', '%s', '80' from \
                    activate_pmid where version = '0' and subject \
                    containing '%s' and object containing '%s'" % \
                    (table, subj, obj, subj, obj)
                try:
                    self.muscopedia.dbcursor.execute(sqlstmt)
                except: pass
            if (count % 1000) == 0:
                self.muscopedia.dbconnect.commit()
                self.PostRun([str(count) + ' interacting pairs

```

```

        processed for ' + table])
        print str(count) + ' interacting pairs processed for '
            + table
        self.PostRun([str(table) + ' table processed'])
        print str(table) + ' table processed'

def commandASbind80(self):
    """
    Invoked on 'asbind80' command. It processes version 0 of bind_pmid
    table to look for SVOs containing entities in job_clone_master table in
    both subject and object attributes. For SVOs whose subject and object
    attributes contain entities in job_clone_master table (abbreviated clone
    names), the attributes were replaced with the abbreviated clone name and
    tagged as version 80.

    Usage: asbind80
    """
    self.muscopedia.dbcursor.execute('select jclone from \
        job_clone_master')
    clone = [clone[0].strip() for clone in
        self.muscopedia.dbcursor.fetchall()]
    count = 0
    for table in ['bind_pmid']:
        for subj in clone:
            for obj in clone:
                count = count + 1
                sqlstmt = "insert into %s (pmid, subject, object, \
                    version) select pmid, '%s', '%s', '80' from \
                    bind_pmid where version = '0' and subject \
                    containing '%s' and object containing '%s'" % \
                    (table, subj, obj, subj, obj)
                try:
                    self.muscopedia.dbcursor.execute(sqlstmt)
                except: pass
            if (count % 1000) == 0:
                self.muscopedia.dbconnect.commit()
                self.PostRun([str(count) + ' interacting pairs \
                    processed for ' + table])
                print str(count) + ' interacting pairs processed for '
                    + table
                self.PostRun([str(table) + ' table processed'])
                print str(table) + ' table processed'

def commandASenhance80(self):
    """
    Invoked on 'asenhance80' command. It processes version 0 of
    enhance_pmid table to look for SVOs containing entities in
    job_clone_master table in both subject and object attributes. For SVOs
    whose subject and object attributes contain entities in job_clone_master
    table (abbreviated clone names), the attributes were replaced with the
    abbreviated clone name and tagged as version 80.

    Usage: asenhance80
    """
    self.muscopedia.dbcursor.execute('select jclone from \
job_clone_master')
    clone = [clone[0].strip() for clone in
self.muscopedia.dbcursor.fetchall()]
    count = 0
    for table in ['enhance_pmid']:
        for subj in clone:
            for obj in clone:
                count = count + 1
                sqlstmt = "insert into %s (pmid, subject, object, \
                    version) select pmid, '%s', '%s', '80' from \
                    enhance_pmid where version = '0' and subject \
                    containing '%s' and object containing '%s'" % \
                    (table, subj, obj, subj, obj)
                try:
                    self.muscopedia.dbcursor.execute(sqlstmt)
                except: pass
            if (count % 1000) == 0:
                self.muscopedia.dbconnect.commit()
                self.PostRun([str(count) + ' interacting pairs \
                    processed for ' + table])
                print str(count) + ' interacting pairs processed for '
                    + table

```

```

        self.PostRun([str(table) + ' table processed'])
        print str(table) + ' table processed'

def commandASdegrade80(self):
    """
    Invoked on 'asdegrade80' command. It processes version 0 of
    degrade_pmid table to look for SVOs containing entities in
    job_clone_master table in both subject and object attributes. For SVOs
    whose subject and object attributes contain entities in job_clone_master
    table (abbreviated clone names), the attributes were replaced with the
    abbreviated clone name and tagged as version 80.

    Usage: asdegrade80
    """
    self.muscopedia.dbcursor.execute('select jclone from \
        job_clone_master')
    clone = [clone[0].strip() for clone in
        self.muscopedia.dbcursor.fetchall()]
    count = 0
    for table in ['degrade_pmid']:
        for subj in clone:
            for obj in clone:
                count = count + 1
                sqlstmt = "insert into %s (pmid, subject, object, \
                    version) select pmid, '%s', '%s', '80' from \
                    degrade_pmid where version = '0' and subject \
                    containing '%s' and object containing '%s'" % \
                    (table, subj, obj, subj, obj)
                try:
                    self.muscopedia.dbcursor.execute(sqlstmt)
                except: pass
                if (count % 1000) == 0:
                    self.muscopedia.dbconnect.commit()
                    self.PostRun([str(count) + ' interacting pairs \
                        processed for ' + table])
                    print str(count) + ' interacting pairs processed for '
                        + table
                    self.PostRun([str(table) + ' table processed'])
                    print str(table) + ' table processed'

def commandASinhibit80(self):
    """
    Invoked on 'asinhibit80' command. It processes version 0 of
    inhibit_pmid table to look for SVOs containing entities in
    job_clone_master table in both subject and object attributes. For SVOs
    whose subject and object attributes contain entities in job_clone_master
    table (abbreviated clone names), the attributes were replaced with the
    abbreviated clone name and tagged as version 80.

    Usage: asinhibit80
    """
    self.muscopedia.dbcursor.execute('select jclone from \
        job_clone_master')
    clone = [clone[0].strip() for clone in
        self.muscopedia.dbcursor.fetchall()]
    count = 0
    for table in ['inhibit_pmid']:
        for subj in clone:
            for obj in clone:
                count = count + 1
                sqlstmt = "insert into %s (pmid, subject, object, \
                    version) select pmid, '%s', '%s', '80' from \
                    inhibit_pmid where version = '0' and subject \
                    containing '%s' and object containing '%s'" % \
                    (table, subj, obj, subj, obj)
                try:
                    self.muscopedia.dbcursor.execute(sqlstmt)
                except: pass
                if (count % 1000) == 0:
                    self.muscopedia.dbconnect.commit()
                    self.PostRun([str(count) + ' interacting pairs \
                        processed for ' + table])
                    print str(count) + ' interacting pairs processed for '
                        + table
                    self.PostRun([str(table) + ' table processed'])
                    print str(table) + ' table processed'

```

```

def commandASppi50(self):
    """
    Invoked on 'asppi50'

    Usage: asppi50
    """
    self.muscopedia.dbcursor.execute('select jclone from \
        job_clone_master')
    clone = [clone[0].strip() for clone in
        self.muscopedia.dbcursor.fetchall()]
    count = 0
    for subj in clone:
        for obj in clone:
            count = count + 1
            sqlstmt = "insert into ppi_pmid (pmid, verb, subject, object,\
                version) select pmid, verb, '%s', '%s', '50' from ml_svo \
                where subject containing '%s' and object containing '%s'" \
                % (subj, obj, subj, obj)
            try:
                self.muscopedia.dbcursor.execute(sqlstmt)
            except: pass
            if (count % 1000) == 0:
                self.muscopedia.dbconnect.commit()
                self.PostRun([str(count) + ' interacting pairs \
                    processed'])
                print str(count) + ' interacting pairs processed'
    self.PostRun([str(table) + ' table processed'])
    print str(table) + ' table processed'

def commandASactivateDistinct(self):
    self.muscopedia.dbcursor.execute("delete from activate_distinct")
    self.muscopedia.dbconnect.commit()
    sqlstmt = "insert into activate_distinct (subject, object) \
        select distinct subject, object from activate_pmid p where \
        p.version=80"
    self.muscopedia.dbcursor.execute(sqlstmt)
    self.muscopedia.dbconnect.commit()

def commandASbindDistinct(self):
    self.muscopedia.dbcursor.execute("delete from bind_distinct")
    self.muscopedia.dbconnect.commit()
    sqlstmt = "insert into bind_distinct (subject, object) \
        select distinct subject, object from bind_pmid p where \
        p.version=80"
    self.muscopedia.dbcursor.execute(sqlstmt)
    self.muscopedia.dbconnect.commit()

def commandBindEDA(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from \
        bind_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    self.muscopedia.dbcursor.execute("delete from bindcount")
    self.muscopedia.dbconnect.commit()
    clone = amap.clean_combination()
    pfile = open('p'+command[1], 'w')
    cfile = open('c'+command[1], 'w')
    for k in clone.keys():
        temp = k.split('--')
        pfile.write(temp[0] + '\t(pp)\t' + temp[1] + ' = bind' +
            os.linesep)
        cfile.write(temp[0] + '\t(pp)\t' + temp[1] + ' = ' + str(clone[k])
            + os.linesep)
        self.muscopedia.dbcursor.execute("insert into bindcount (clone1, \
            clone2, occur) values (?, ?, ?)", (temp[0], temp[1], str(clone[k])))
    self.muscopedia.dbconnect.commit()
    pfile.close()

def commandBindSIF(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from \
        bind_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    clone = amap.clean_combination()
    filename = open(command[1], 'w')

```



```

for k in clone.keys():
    temp = k.split('--')
    filename.write(temp[0] + '\tpp\t' + temp[1])
filename.close()

def commandCooccurrenceEDA(self, command):
    sqlstmt = "select unigenel, unigene2, paircount from \
        pubgene_as_unigene"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = str(command[1])
    clone = [[clone[0].strip(), clone[1].strip(), str(clone[2])]
        for clone in self.muscopedia.dbcursor.fetchall()]
    filename = open(filename, 'w')
    filename.write('NumberOfCooccurrence' + os.linesep)
    for c in clone: filename.write(c[0] + '\t(pp)\t' + c[1] + ' = ' + c[2]
        + os.linesep)
    filename.close()

def commandCWcoexpLoad(self, command):
    f = open(command[1], 'r')
    self.muscopedia.dbcursor.execute("delete from cw_inv_coexp")
    self.muscopedia.dbconnect.commit()
    count = 0
    for line in f:
        line = line.split('\t')
        self.muscopedia.dbcursor.execute("insert into cw_inv_coexp \
            (probel, probe2, correlation) values ('%s', '%s', '%3.3f')" \
            % (line[1], line[2], float(line[3][: -1])))
        count = count + 1
        if (count % 10000) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' correlations processed'])
            print str(count) + ' correlations processed'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' correlations processed'])
    print str(count) + ' correlations processed'

def commandDegradeEDA(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from \
        degrade_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    self.muscopedia.dbcursor.execute("delete from degradecount")
    self.muscopedia.dbconnect.commit()
    clone = amap._clean_combination()
    pfile = open('p'+command[1], 'w')
    cfile = open('c'+command[1], 'w')
    for k in clone.keys():
        temp = k.split('--')
        pfile.write(temp[0] + '\t(pp)\t' + temp[1] + ' = degrade' +
            os.linesep)
        cfile.write(temp[0] + '\t(pp)\t' + temp[1] + ' = ' + str(clone[k])
            + os.linesep)
        self.muscopedia.dbcursor.execute("insert into degradecount \
            (clone1, clone2, occur) values (?, ?, ?)",
            (temp[0], temp[1], str(clone[k])))
    self.muscopedia.dbconnect.commit()
    pfile.close()

def commandDegradeSIF(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from \
        degrade_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    clone = amap._clean_combination()
    filename = open(command[1], 'w')
    for k in clone.keys():
        temp = k.split('--')
        filename.write(temp[0] + '\tpp\t' + temp[1])
    filename.close()

def commandCWcoexpSIF(self, command):
    sqlstmt = "select unigenel, unigene2 from cw_coexp_as_unigene where \
        correlation > 0.75 or correlation < -0.75"
    self.muscopedia.dbcursor.execute(sqlstmt)

```

```

filename = open(str(command[1]), 'w')
for clone in self.muscopedia.dbcursor:
    try: filename.write(clone[0].strip() + '\tpp\t' + clone[1].strip()
        + os.linesep)
    except: pass
filename.close()

def commandEnhanceEDA(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from
enhance_pmid \
    where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    self.muscopedia.dbcursor.execute("delete from bindcount")
    self.muscopedia.dbconnect.commit()
    clone = amap._clean_combination()
    pfile = open('p'+command[1], 'w')
    cfile = open('c'+command[1], 'w')
    for k in clone.keys():
        temp = k.split('--')
        pfile.write(temp[0] + '\t(pp)\t' + temp[1] + ' = enhance' +
            os.linesep)
        cfile.write(temp[0] + '\t(pp)\t' + temp[1] + ' = ' + str(clone[k])
            + os.linesep)
        self.muscopedia.dbcursor.execute("insert into enhancecount \
            (clone1, clone2, occur) values (?, ?, ?)",
            (temp[0], temp[1], str(clone[k])))
    self.muscopedia.dbconnect.commit()
    pfile.close()

def commandEnhanceSIF(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from \
        enhance_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    clone = amap._clean_combination()
    filename = open(command[1], 'w')
    for k in clone.keys():
        temp = k.split('--')
        filename.write(temp[0] + '\tpp\t' + temp[1])
    filename.close()

def commandDiseaseProtein(self, command):
    """
    Invoked on 'dispro' command. Using clones in job_clone_master table,
    it looks for SVOs in ml_svo table for either the subject attribute
    containing clones and object attribute containing the disease term
    or subject containing the disease in question and object
    containing the clones.

    Usage: dispro <disease name>
    """
    self.muscopedia.dbcursor.execute('select jclone from
job_clone_master')
    clone = [clone[0].strip() for clone in
self.muscopedia.dbcursor.fetchall()]
    count = 0
    if len(command[1:]) > 1: disease_name = str(' '.join(command[1:]))
    else: disease_name = str(command[1])
    for protein in clone:
        count = count + 1
        sqlstmt = "insert into protein_disease (pmid, verb, disease, \
            protein) select pmid, verb, '%s', '%s' from ml_svo where \
            (subject containing '%s' and object containing '%s') or \
            (subject containing '%s' and object containing '%s')" \
            % (disease_name, protein, disease_name, protein, \
            disease_name)
        try:
            self.muscopedia.dbcursor.execute(sqlstmt)
        except IOError: pass
    if (count % 50) == 0:
        self.muscopedia.dbconnect.commit()
        self.PostRun([str(count) + ' protein-disease pairs processed \
            for ' + disease_name])
        print str(count) + ' protein-disease pairs processed for ' +

```

```

        disease_name
    self.PostRun([disease_name + ' processed'])
    print disease_name + ' processed'

def commandDM_2entityinteraction(self, command):
    entity1 = ''.join(command[1:]).split(';')[0]
    entity2 = ''.join(command[1:]).split(';')[1]
    sqlstmt = "select distinct verb from ml_svo where (subject \
        containing '%s' and object containing '%s') or (subject \
        containing '%s' and object containing '%s')" % (entity1, \
        entity2, entity2, entity1)
    self.muscopedia.dbcursor.execute(sqlstmt)
    interactionverbs = [verb[0].strip() for verb in
        self.muscopedia.dbcursor.fetchall()]
    sqlstmt = "select distinct pmid from ml_svo where (subject \
        containing '%s' and object containing '%s') or (subject \
        containing '%s' and object containing '%s')" % (entity1, \
        entity2, entity2, entity1)
    self.muscopedia.dbcursor.execute(sqlstmt)
    interactionpmid = [pmid[0].strip() for pmid in
        self.muscopedia.dbcursor.fetchall()]
    sqlstmt = "select pmid from pmc_abstract where text containing '%s' \
        and text containing '%s' union select pmid from \
        text_replace_abstract where \
        text containing '%s' and text containing '%s'" \
        % (entity1, entity2, entity1, entity2)

    self.muscopedia.dbcursor.execute(sqlstmt)
    cooccurpmid = [str(pmid[0]).strip() for pmid in
        self.muscopedia.dbcursor.fetchall()]
    cooccuronly = [pmid for pmid in cooccurpmid
        if pmid not in interactionpmid]
    print "Interactions between %s and %s: %s" % (entity1, entity2,
        str(interactionverbs))
    print "These interactions are found in the following PMIDs: %s" %
        (str(interactionpmid))
    print "PMIDs with co-occurrence but no interactions found: %s" %
        (str(cooccuronly))
    return (interactionverbs, interactionpmid, cooccurpmid, cooccuronly)

def commandGenerateMicroarrayCorrelationIntersects(self):
    self.muscopedia.dbcursor.execute('delete from pnts_intersect')
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.execute("insert into pnts_intersect \
        select distinct t.unigenel, t.unigene2 from \
        ts_coexp_as_unigene t join pn_coexp_as_unigene p \
        on t.unigenel=p.unigenel and t.unigene2=p.unigene2")
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.execute('delete from pntscw_intersect')
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.execute("insert into pntscw_intersect \
        select distinct p.unigenel, p.unigene2 from \
        pnts_intersect p join cw_coexp_as_unigene c \
        on p.unigenel=c.unigenel and p.unigene2=c.unigene2")
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.execute('delete from pntscwlc_intersect')
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.execute("insert into pntscwlc_intersect \
        select distinct p.unigenel, p.unigene2 from \
        pntscw_intersect p join lc_coexp_as_unigene l on \
        p.unigenel=l.unigenel and p.unigene2=l.unigene2")
    self.muscopedia.dbconnect.commit()

def commandGenerateMicroarrayCorrelationUnion(self):
    self.muscopedia.dbcursor.execute("delete from all_coexp")
    self.muscopedia.dbcursor.execute("create table all_coexp_t \
        (unigenel char(12), unigene2 char(12))")
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.execute("insert into all_coexp_t select \
        unigenel, unigene2 from ts_coexp_as_unigene")
    self.muscopedia.dbcursor.execute("insert into all_coexp_t select \
        unigenel, unigene2 from pn_coexp_as_unigene")
    self.muscopedia.dbcursor.execute("insert into all_coexp_t select \
        unigenel, unigene2 from cw_coexp_as_unigene")
    self.muscopedia.dbcursor.execute("insert into all_coexp_t select \
        unigenel, unigene2 from lc_coexp_as_unigene")

```

```

self.muscopedia.dbconnect.commit()
self.muscopedia.dbcursor.execute("insert into all_coexp select \
distinct unigenel, unigene2 from all_coexp_t")
self.muscopedia.dbconnect.commit()
self.muscopedia.dbcursor.execute("drop table all_coexp_t")
self.muscopedia.dbconnect.commit()

def commandGenerateMicroarrayExpStatistics(self):
self.muscopedia.dbcursor.execute("delete from ma_statistics_bt")
self.muscopedia.dbconnect.commit()
self.muscopedia.dbcursor.execute("select distinct unigene from \
ts_array_data")
clones = self.muscopedia.dbcursor.fetchall()
count = 0
for clone in clones:
data = []
sqlstmt = "select V10_A, V10_B, V10_C, V12_A, V12_B, V12_C, P1_A, \
P1_B, P1_C, P2_A, P2_B, P2_C, P3_A, P3_B, P3_C, P8_5_A, \
P8_5_B, P8_5_C, P12_5_A, P12_5_B, P12_5_C, P14_5_A, \
P14_5_B, P14_5_C, P17_5_A, P17_5_B, P17_5_C, Lac1_A, \
Lac1_B, Lac1_C, Lac3_A, Lac3_B, Lac3_C, Lac7_A, Lac7_B, \
Lac7_C, Inv1_A, Inv1_B, Inv1_C, Inv2_A, Inv2_B, Inv2_C, \
Inv3_A, Inv3_B, Inv3_C, Inv4_A, Inv4_B, Inv4_C, Inv20_A, \
Inv20_B, Inv20_C from ts_array_data where \
unigene = '%s' % clone

try:
self.muscopedia.dbcursor.execute(sqlstmt)
exp = self.muscopedia.dbcursor.fetchall()
combination = UCFunctions.xuniqueCombinations(exp[0], 2)
data = data + [c[0]/c[1] for c in combination]
except: pass
sqlstmt = "select p.DF40_1, p.DF40_2, p.DF40_3, p.DF8_1, p.DF8_2, \
p.DF8_3, p.DF16_1, p.DF16_2, p.DF16_3, p.NFD2lact1, \
p.NFD2lact2, p.NFD2lact3, p.NFD2lact4, p.NFD9lact1, \
p.NFD9lact2, p.NFD9lact3, p.NFD9lact4, p.NFD12Preg1, \
p.NFD12Preg2, p.NFD12Preg3, p.NFD12Preg4, p.NFD1lact1, \
p.NFD1lact2, p.NFD1lact3, p.NFD1lact4, p.NFD17Preg1, \
p.NFD17Preg2, p.NFD17Preg3, p.NFD17Preg4 from \
pn_array_data p join affy_annot a on \
p.probeid = a.probeid where a.unigene = '%s' % clone

try:
self.muscopedia.dbcursor.execute(sqlstmt)
exp = self.muscopedia.dbcursor.fetchall()
combination = UCFunctions.xuniqueCombinations(exp[0], 2)
data = data + [c[0]/c[1] for c in combination]
except: pass
try:
btsample = [UCFunctions.ArithmeticMean
(UCFunctions.sample_wr(data, 300))
for i in xrange(500)]
mean = UCFunctions.ArithmeticMean(btsample)
sd = UCFunctions.standarddeviation(btsample, mean)
self.muscopedia.dbcursor.execute("insert into \
ma_statistics_bt (unigene, mean, sd) values \
('%s', '%3.3f', '%3.3f') % (clone[0], mean, sd))
except: pass
count = count + 1
if (count % 1000) == 0:
self.muscopedia.dbconnect.commit()
self.PostRun([str(count) + ' unigenes processed'])
print str(count) + ' unigenes processed'
self.muscopedia.dbconnect.commit()
self.PostRun([str(count) + ' unigenes processed'])
print str(count) + ' unigenes processed'

def commandGenerateNonMicroarrayCache(self):
self.muscopedia.dbcursor.execute('delete from pubgene_cache')
self.muscopedia.dbconnect.commit()
self.muscopedia.dbcursor.execute("insert into pubgene_cache select \
unigenel, unigene2, paircount, clonel, clone2 from \
pubgene_as_unigene")
self.muscopedia.dbconnect.commit()

def commandInhibitEDA(self, command):
from conceptmap import mapmaker
amap = mapmaker.mapmaker()
self.muscopedia.dbcursor.execute("select subject, object from \

```

```

        inhibit_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    self.muscopedia.dbcursor.execute("delete from inhibitcount")
    self.muscopedia.dbconnect.commit()
    clone = amap._clean_combination()
    pfile = open('p'+command[1], 'w')
    cfile = open('c'+command[1], 'w')
    for k in clone.keys():
        temp = k.split('--')
        pfile.write(temp[0] + '\t(pp)\t' + temp[1] + ' = inhibit' +
            os.linesep)
        cfile.write(temp[0] + '\t(pp)\t' + temp[1] + ' = ' + str(clone[k])
            + os.linesep)
        self.muscopedia.dbcursor.execute("insert into inhibitcount \
            (clone1, clone2, occur) values (?, ?, ?)", (temp[0], temp[1],
                str(clone[k])))
    self.muscopedia.dbconnect.commit()
    pfile.close()

def commandInhibitsSIF(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from \
        inhibit_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    clone = amap._clean_combination()
    filename = open(command[1], 'w')
    for k in clone.keys():
        temp = k.split('--')
        filename.write(temp[0] + '\tpp\t' + temp[1])
    filename.close()

def commandJobCloneOccurrence(self):
    """
    Invoked on 'jcloneoccur' command. It counts the number of abstracts
    with the entity names (from job_clone_master table) and logs it into
    jclone_occurrence table.

    Usage: jcloneoccur
    """
    self.muscopedia.dbcursor.execute('select jclone from \
        job_clone_master')
    clone = [clone[0].strip() for clone in
        self.muscopedia.dbcursor.fetchall()]
    self.muscopedia.dbcursor.execute("delete from jclone_occurrence")
    count = 0
    for obj in clone:
        sqlstmt = "insert into jclone_occurrence (jclone, occurrence) \
            select '%s', count(distinct pmid) from text_replace_abstract \
            where text containing '%s'" % (obj, obj)
        self.muscopedia.dbcursor.execute(sqlstmt)
        count = count + 1
        if (count % 100) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' clones processed'])
            print str(count) + ' clones processed'
    self.PostRun([str(table) + ' clones processed'])
    print str(table) + ' clones processed'

def commandJobCloneOccurrenceIndividual(self):
    """
    Invoked on 'jcloneoccurind' command. It tabulates a list of PMIDs with
    abstracts with entity names (from job_clone_master table) in it into
    jclone_occurrence_pmid table. This function differs from
    commandJobCloneOccurrence() in that actual PMIDs are listed in this
    function.

    Usage: jcloneoccurind
    """
    self.muscopedia.dbcursor.execute('select jclone from \
        job_clone_master')
    clones = [clone[0].strip() for clone in
        self.muscopedia.dbcursor.fetchall()]
    self.muscopedia.dbcursor.execute("delete from jclone_occurrence_pmid")
    count = 0
    for clone in clones:
        count = count + 1

```

```

        sqlstmt = "insert into jclone_occurrence_pmid (pmid, clone) \
                    select pmid, '%s' from text_replace_abstract where \
                    text containing '%s'" % (clone, clone)
    try:
        self.muscopedia.dbcursor.execute(sqlstmt)
    except: pass
    if (count % 100) == 0:
        self.muscopedia.dbconnect.commit()
        self.PostRun([str(count) + ' clones for
                        jclone_occurrence_pmid'])
        print str(count) + ' clones processed for
                        jclone_occurrence_pmid'
    self.PostRun(['jclone_occurrence_pmid table processed'])
    print 'jclone_occurrence_pmid table processed'

def commandJobCloneOccurrencePoisson(self):
    """
    Invoked on 'jcloneoccurpoisson'.

    Usage: jcloneoccurpoisson
    """
    from conceptmap import names
    poisson = names.Poisson()
    self.muscopedia.dbcursor.execute('select count(pmid) from \
                                      pmc_abstract')
    abstractcount = float(self.muscopedia.dbcursor.fetchall()[0][0])
    self.muscopedia.dbcursor.execute('select jclone, occurrence from \
                                      jclone_occurrence')
    dataset = [[clone[0].strip(), clone[1]] for clone in
                self.muscopedia.dbcursor.fetchall()]
    self.muscopedia.dbcursor.execute("delete from jclone_occur_stat")
    count = 0
    for subj in dataset:
        for obj in dataset:
            mean = (float(subj[1])/abstractcount)*
                    (float(obj[1])/abstractcount)
            poisson.mean = mean
            (poi95, prob) = poisson.inverseCDF(0.95)
            (poi99, prob) = poisson.inverseCDF(0.99)
            count = count + 1
            sqlstmt = "insert into jclone_occur_stat (clone1, clone2, \
                randomoccur, poisson95, poisson99) values \
                ('%s', '%s', '%.6f', '%s', '%s')" % (str(subj[0]),
                str(obj[0]), mean, str(poi95), str(poi99))
            try:
                self.muscopedia.dbcursor.execute(sqlstmt)
            except IOError: pass
            if (count % 1000) == 0:
                self.muscopedia.dbconnect.commit()
                self.PostRun([str(count) + ' interacting pairs processed
                                for jclone_occur_stat'])
                print str(count) + ' interacting pairs processed for
                                jclone_occur_stat'
            self.PostRun(['jclone_occur_stat table processed'])
            print 'jclone_occur_stat table processed'

def commandJobCloneLoad(self, command):
    """
    Invoked on 'jobcloneload' command. It processes the file and load the
    data into job_clone_map table. After which, a distinct set of clones is
    loaded into job_clone_master table. The input file's format is {<clone
    name, ID, ID type>}, separated by '\t'. This command is used to load a
    list of proteins/genes entities (clones) into the system for linking
    into concept maps.

    Usage: jobcloneload <filename>
    """
    data = [(clone.split('\t')[0].upper(), clone.split('\t')[1],
              clone.split('\t')[2][:-1]) for clone in open(command[1],
              'r').readlines()[:-1]]
    print data[10]
    print 'Number of clones in ' + command[1] + ' file: ' + str(len(data))
    count = 0
    for clone in data:
        count = count + 1
    try:
        self.muscopedia.dbcursor.execute('insert into job_clone_map \

```

```

        (jclone, other_id, idtype) values (?, ?, ?)', clone)
    except: pass
    if (count % 1000) == 0:
        self.muscopedia.dbconnect.commit()
        self.PostRun([str(count) + ' clones processed for
            job_clone_map table'])
        print str(count) + ' clones processed for job_clone_map table'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' clones processed for job_clone_map
        table'])
    print str(count) + ' clones processed for job_clone_map table'
    self.muscopedia.dbcursor.execute('delete from job_clone_master')
    sqlstmt = 'insert into job_clone_master (jclone) \
        select distinct jclone from job_clone_map'
    self.muscopedia.dbcursor.execute(sqlstmt)
    self.muscopedia.dbconnect.commit()

def commandLCcoexpLoad(self, command):
    f = open(command[1], 'r')
    self.muscopedia.dbcursor.execute("delete from mdata_correlation")
    self.muscopedia.dbconnect.commit()
    count = 0
    for line in f:
        line = line.split('\t')
        self.muscopedia.dbcursor.execute("insert into mdata_correlation \
            (clone_id1, clone_id2, correlation) values ('%s', '%s', \
            '%3.3f')" % (line[1], line[2], float(line[3][:-1])))
        count = count + 1
        if (count % 10000) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' correlations processed'])
            print str(count) + ' correlations processed'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' correlations processed'])
    print str(count) + ' correlations processed'

def commandLCcoexpSIF(self, command):
    sqlstmt = "select unigenel, unigene2 from lc_coexp_as_unigene where \
        correlation > 0.75 or correlation < -0.75"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = open(str(command[1]), 'w')
    for clone in self.muscopedia.dbcursor:
        try: filename.write(clone[0].strip() + '\t' + clone[1].strip()
            + os.linesep)
        except: pass
    filename.close()

def commandLinkAll(self, command):
    """
    Invoked on 'linkall' command. Activates 'linkmuscopedia', 'linkpubmed'
    and 'linkmontylingua' command.

    Usage: linkall
    """
    self.commandLinkMuscopediaDB(command)
    self.commandLinkPubMed()
    self.commandLinkMontyLingua()

def commandKMglucocorticoid1(self):
    self.muscopedia.dbcursor.execute("select unigenecol, unigene, \
        SIFP12, SIP12 from km_mouse_vitro_id")
    data = self.muscopedia.dbcursor.fetchall()
    calc = [(x[0], x[1], float(x[2])/float(x[3]))
        for x in data] # log(SIFP12)/log(SIP12)
    self.muscopedia.dbcursor.execute("delete from km_glucocorticoid1")
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.executemany("insert into km_glucocorticoid1 \
        (unigenecol, unigene, ratio) values (?, ?, ?)", calc)
    self.muscopedia.dbconnect.commit()

def commandKMglucocorticoid1out(self, command):
    self.muscopedia.dbcursor.execute("select k.unigene, k.symbol, \
        k.SIFP12, k.SIP12, r.ratio from km_mouse_vitro_id k join \
        km_glucocorticoid1 r on k.unigene=r.unigene")
    data = self.muscopedia.dbcursor.fetchall()
    f = open(command[1], 'w')

```

```

f.write('UniGene\tSymbol\tSIFP12\tSIP12\tRatio' + os.linesep)
for x in data: f.write('%s\t%s\t%s\t%s\t%s' %
                        (x[0], str(x[1]).strip(), str(x[2]),
                         str(x[3]), str(x[4])) + os.linesep)

f.close()

def commandKMinInsulin1(self):
    self.muscopedia.dbcursor.execute("select unigenecluster, unigene, \
        SIFP12, SFP12 from km_mouse_vitro_id")
    data = self.muscopedia.dbcursor.fetchall()
    calc = [(x[0], x[1], float(x[2])/float(x[3]))
             for x in data] # log(SIFP12)/log(SFP12)
    self.muscopedia.dbcursor.execute("delete from km_insulin1")
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.executemany("insert into km_insulin1 \
        (unigenecluster, unigene, ratio) values (?, ?, ?)", calc)
    self.muscopedia.dbconnect.commit()

def commandKMinInsulin1out(self, command):
    self.muscopedia.dbcursor.execute("select k.unigene, k.symbol, \
        k.SIFP12, k.SFP12, r.ratio from km_mouse_vitro_id k join \
        km_insulin1 r on k.unigene=r.unigene")
    data = self.muscopedia.dbcursor.fetchall()
    f = open(command[1], 'w')
    f.write('UniGene\tSymbol\tSIFP12\tSFP12\tRatio' + os.linesep)
    for x in data: f.write('%s\t%s\t%s\t%s\t%s' %
                           (x[0], str(x[1]).strip(), str(x[2]),
                            str(x[3]), str(x[4])) + os.linesep)

    f.close()

def commandKMprolactin1(self):
    self.muscopedia.dbcursor.execute("select unigenecluster, unigene, \
        SIFP12, SIF12 from km_mouse_vitro_id")
    data = self.muscopedia.dbcursor.fetchall()
    calc = [(x[0], x[1], float(x[2])/float(x[3]))
             for x in data] # log(SIFP12)/log(SFP12)
    self.muscopedia.dbcursor.execute("delete from km_prolactin1")
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.executemany("insert into km_prolactin1 \
        (unigenecluster, unigene, ratio) values (?, ?, ?)", calc)
    self.muscopedia.dbconnect.commit()

def commandKMprolactin1out(self, command):
    self.muscopedia.dbcursor.execute("select k.unigene, k.symbol, \
        k.SIFP12, k.SIF12, r.ratio from km_mouse_vitro_id k join \
        km_prolactin1 r on k.unigene=r.unigene")
    data = self.muscopedia.dbcursor.fetchall()
    f = open(command[1], 'w')
    f.write('UniGene\tSymbol\tSIFP12\tSIF12\tRatio' + os.linesep)
    for x in data: f.write('%s\t%s\t%s\t%s\t%s' %
                           (x[0], str(x[1]).strip(), str(x[2]),
                            str(x[3]), str(x[4])) + os.linesep)

    f.close()

def commandLinkMontyLingua(self):
    """
    Invoked on 'linkmontylingua' command. It initiates an instance of
    montylingua.MontyLingua. MontyLingua() and brings it online.

    Usage: linkmontylingua"""
    self.montylingua = montylingua.MontyLingua.MontyLingua()
    self.result.append("MontyLingua Online")

def commandLinkMuscopediaDB(self, command):
    """
    Invoked on 'linkmuscopedia' command. It calls for
    PubMedGrabber.FBUtilities.__init__() to connect to Muscopedia Database.

    Usage: linkmuscopedia
           linkmuscopedia <connection term>

    <connection term>: full path of the Firebird database, including host.
                       It should take the form of '<host name>:<full path
                       name>:<database user name>:<database user password>'.
    If no connection term is given, then the default connection term is
    used."""
    if len(command) == 1:

```



```

        command.append('localhost:' +
            os.path.join(self.environment["inidir"], 'BioDatabases' +
                os.sep + 'muscopedia.fdb') + ':mouse:mouse')
print "Connecting to Muscopedia Database"
print "Connection: " + str(command[1])
self.result.append("Connecting to Muscopedia Database")
self.result.append("Connection: " + str(command[1]))
self.PostRun(self.result)
del self.result[:]
self.muscopedia = PubMedGrabber.FBUtilities(str(command[1]))
self.result.append("Muscopedia Database Online")

def commandLoadAccUGLL(self, command):
    data = open(command[1], 'r').readlines()
    data = data[0].split('\r')[1:-1]
    data = [x.split('\t') for x in data]
    data = [(x[0], x[1], x[2]) for x in data]
    self.muscopedia.dbcursor.executemany("insert into acc_ug_ll \
        (accession, unigene, locuslink) values (?, ?, ?)", data)
    self.muscopedia.dbconnect.commit()

def commandLoadAffyAnnt(self, command):
    data = open(command[1], 'r').readlines()
    data = [x[:-1] for x in data]
    data = [x.split('\t') for x in data]
    data = data[1:]
    t1 = []
    t2 = []
    for x in data:
        if len(x[0]) > 35: x[0] = x[0][:35] # probeid
        elif len(x[2]) > 12: x[2] = x[2][:12] # unigenecluster
        elif len(x[3]) > 12: x[3] = x[3][:12] # unigene
        elif len(x[4]) > 128: x[4] = x[4][:128] # description
        elif len(x[5]) > 40: x[5] = x[5][:35] # symbol
        elif len(x[6]) > 30: x[6] = x[6][:30] # ensembl
        elif len(x[7]) > 30: x[7] = x[7][:30] # entrez
        t1 = (x[0], x[2], x[3], x[5], x[7], x[4], x[6])
    try:
        self.muscopedia.dbcursor.execute("insert into affy_annot \
            (probeid, unigenecluster, unigene, symbol, entrez, \
                description, ensembl) values (?, ?, ?, ?, ?, ?, ?)", t1)
    except: print t1
    self.muscopedia.dbconnect.commit()

def commandLoadKMdata(self, command):
    data = open(command[1], 'r').readlines()
    data = [x[:-1] for x in data]
    data = [x.split('\t') for x in data]
    data = data[1:]
    self.muscopedia.dbcursor.execute("delete from km_mouse_vitro")
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.executemany("insert into km_mouse_vitro \
        (probeid, sFP12, sIF12, sIFP12, s8, sIFPG10, sIFPG12, s12, \
            sIP10, sFP10, sIF10, sIFP10, sIFG10, sIFG12, s10, sIP12) values \
            (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", data)
    self.muscopedia.dbconnect.commit()

def commandLoadPNdata(self, command):
    data = open(command[1], 'r').readlines()
    data = [x[:-1] for x in data]
    data = [x.split('\t') for x in data]
    data = data[1:]
    self.muscopedia.dbcursor.execute("delete from pn_array_data")
    self.muscopedia.dbconnect.commit()
    for line in data:
        sqlstmt = "insert into pn_array_data (probeid, DF40_1, DF40_2, \
            DF40_3, DF8_1, DF8_2, DF8_3, DF16_1, DF16_2, DF16_3, \
            NFD2lact1, NFD2lact2, NFD2lact3, NFD2lact4, NFD9lact1, \
            NFD9lact2, NFD9lact3, NFD9lact4, NFD12Preg1, NFD12Preg2, \
            NFD12Preg3, NFD12Preg4, NFD1lact1, NFD1lact2, NFD1lact3, \
            NFD1lact4, NFD17Preg1, NFD17Preg2, NFD17Preg3, \
            NFD17Preg4) values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, \
                ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"
        try: self.muscopedia.dbcursor.execute(sqlstmt, line)
        except: print line
    self.muscopedia.dbconnect.commit()

```

```

def commandLoadTSdata(self, command):
    data = open(command[1], 'r').readlines()
    data = [x[:-1] for x in data]
    data = [x.split('\t') for x in data]
    data = data[1:]
    self.muscopedia.dbcursor.execute("delete from ts_array_data")
    self.muscopedia.dbconnect.commit()
    sqlstmt = "insert into ts_array_data (unigene, V10_A, V10_B, V10_C, \
        V12_A, V12_B, V12_C, P1_A, P1_B, P1_C, P2_A, P2_B, P2_C, P3_A, \
        P3_B, P3_C, P8_5_A, P8_5_B, P8_5_C, P12_5_A, P12_5_B, P12_5_C, \
        P14_5_A, P14_5_B, P14_5_C, P17_5_A, P17_5_B, P17_5_C, Lac1_A, \
        Lac1_B, Lac1_C, Lac3_A, Lac3_B, Lac3_C, Lac7_A, Lac7_B, Lac7_C, \
        Inv1_A, Inv1_B, Inv1_C, Inv2_A, Inv2_B, Inv2_C, Inv3_A, Inv3_B, \
        Inv3_C, Inv4_A, Inv4_B, Inv4_C, Inv20_A, Inv20_B, Inv20_C) \
        values (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, \
        ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
    self.muscopedia.dbcursor.executemany(sqlstmt, data)
    self.muscopedia.dbconnect.commit()

def commandMakePGNameMap(self, command):
    from conceptmap import names
    names = names.nameStatistics()
    self.muscopedia.dbcursor.execute('select name1, name2 from pg_comb')
    names.pgcombination = self.muscopedia.dbcursor.fetchall()
    names.writeNameStatisticsGraphFile(command[1])

def commandMakeActivationMap(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from \
        activate_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    amap.writeGraphFile(command[1])

def commandMakeActivationMapSIF(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    sqlstmt = "select distinct s_unigene, o_unigene from \
        activate_as_unigene where s_unigene!=o_unigene"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = str(command[1])
    clone = [[clone[0].strip(), clone[1].strip()] for clone in
        self.muscopedia.dbcursor.fetchall()]
    filename = open(filename, 'w')
    for c in clone: filename.write(c[0] + '\tpp\t' + c[1] + os.linesep)
    filename.close()

def commandMakeBindingMap(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    self.muscopedia.dbcursor.execute("select subject, object from \
        bind_pmid where version = '80'")
    amap.combination = self.muscopedia.dbcursor.fetchall()
    amap.writeGraphFile(command[1])

def commandMakeBindingMapSIF(self, command):
    from conceptmap import mapmaker
    amap = mapmaker.mapmaker()
    sqlstmt = "select distinct s_unigene, o_unigene from bind_as_unigene \
        where s_unigene!=o_unigene"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = str(command[1])
    clone = [[clone[0].strip(), clone[1].strip()] for clone in
        self.muscopedia.dbcursor.fetchall()]
    filename = open(filename, 'w')
    for c in clone: filename.write(c[0] + '\tpp\t' + c[1] + os.linesep)
    filename.close()

def commandMakeBindingMapFromListExact(self, command):
    """
    Invoked on 'makebindingmapfromlistexact' command. It performs a match
    between the given list (input file) and job clone_master table.
    Successful matches were used to find binding interactions in
    bind_pmid table, version 80. This function does 1-skip and direct
    interactions.
    It generates 3 files:
    1. .ser == result file of the match between desired set and

```

2. .dot == extracted interactions in DOT format.
3. .cob == number of multiple extractions per interaction (NLP)

Usage: makebindingmapfromlistexact <input file> <output file>  
 """

```

from conceptmap import mapmaker
amap = mapmaker.mapmaker()
# compare clone list
ifile = open(command[1], 'r').readlines()
ifile = [line[:-1].strip() for line in ifile]
match = {}
for line in ifile:
    sqlstmt = "select jclone from job_clone_master where \
                jclone = '%s'" % (line)
    self.muscopedia.dbcursor.execute(sqlstmt)
    result = self.muscopedia.dbcursor.fetchall()
    if len(result) == 0: match[line] = 'notfound'
    else: match[line] = 'exact'
ofile = open(command[2] + '.ser', 'w')
for key in match.keys():
    ofile.write('\t'.join([key, match[key], os.linesep]))
    if match[key] == 'notfound': del match[key]
ofile.close()
# extract data from bind_pmid
clone = match.keys()
combination = []
for subj in clone:
    for obj in clone:
        sqlstmt = "select subject, object from bind_pmid where \
                    version = 80 and subject = '%s' or (version = 80 and \
                    object = '%s')" % (subj, obj)
        self.muscopedia.dbcursor.execute(sqlstmt)
        combination = combination +
self.muscopedia.dbcursor.fetchall()
# make map
amap.combination = combination
amap.writeGraphFile(command[2] + '.dot')
combination = amap.combination
ofile = open(command[2] + '.cob', 'w')
for key in combination.keys():
    ofile.write('\t'.join([key, str(combination[key]), os.linesep]))
ofile.close()

def commandMicroarrayClonesLoad(self, command):
    """
    Invoked on 'macloneload' command. It processes the input file and load
    into microarray_clone_master table, which contains clone IDs and
    description of the clones for a microarray slide.
    The input file format is {<clone ID, clone description>}, separated
    by '\t'.

    Usage: macloneload <filename>
    """
    data = [(clone.split('\t')[0], clone.split('\t')[1][:-1])
             for clone in open(command[1], 'r').readlines()[1:-1]]
    print 'Number of clones in ' + command[1] + ' file: ' + str(len(data))
    count = 0
    for clone in data:
        count = count + 1
        try:
            self.muscopedia.dbcursor.execute('insert into \
                microarray_clone_master (clone_id, description) values \
                (?,?)', clone)
        except: pass
    if (count % 1000) == 0:
        self.muscopedia.dbconnect.commit()
        self.PostRun([str(count) + ' clones processed for \
            microarray_clone_master table'])
        print str(count) + ' clones processed for \
            microarray_clone_master table'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' clones processed for \
        microarray_clone_master table'])
    print str(count) + ' clones processed for microarray_clone_master \
        table'

```

```

def commandMicroarrayClonesIDLoad(self, command):
    """
    Invoked on 'macloneidload' command. This command is used to load
    alternative IDs for any microarray clone IDs. It processes the input
    file and loads the data into microarray_clone_map table. The input file
    format is {<original microarray clone ID, alternative clone ID, ID
    type>}, separated by '\t'.

    Usage: macloneidload <filename>
    """
    data = [(clone.split('\t')[0], clone.split('\t')[1],
              clone.split('\t')[2][:-1])
             for clone in open(command[1], 'r').readlines()[1:-1]]
    print 'Number of clones in ' + command[1] + ' file: ' + str(len(data))
    count = 0
    for clone in data:
        count = count + 1
        try:
            self.muscopedia.dbcursor.execute('insert into \
            microarray_clone_map (clone_id, other_id, idtype) values \
            (?, ?, ?)', clone)
        except: pass
    if (count % 1000) == 0:
        self.muscopedia.dbconnect.commit()
        self.PostRun([str(count) + ' clones processed for \
        microarray_clone_map table'])
        print str(count) + ' clones processed for microarray_clone_map \
        table'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' clones processed for microarray_clone_map \
    table'])
    print str(count) + ' clones processed for microarray_clone_map table'

def commandMicroarrayCorrelation(self):
    from Scientific import Statistics
    self.muscopedia.dbcursor.execute('select distinct clone_id from \
    microarray_data_abs')
    count = 0
    clonelist = [clone[0].strip() for clone in
                  self.muscopedia.dbcursor.fetchall()]
    for clone1 in clonelist:
        self.muscopedia.dbcursor.execute("select intensity from \
        microarray_data_abs where clone_id = '%s'" % clone1)
        c1data = self.muscopedia.dbcursor.fetchall()
        for clone2 in clonelist:
            self.muscopedia.dbcursor.execute("select intensity from \
            microarray_data_abs where clone_id = '%s'" % clone2)
            c2data = self.muscopedia.dbcursor.fetchall()
            correlation = Statistics.correlation(c1data, c2data)
            count = count + 1
            print 'count: %s %s' % (count, correlation)
            self.muscopedia.dbcursor.execute("insert into \
            mdata_correlation (clone_id1, clone_id2, correlation) \
            values ('%s', '%s', '%3.3f')" % \
            (clone1, clone2, correlation[0]))
            if (count % 1000) == 0:
                self.muscopedia.dbconnect.commit()
                self.PostRun([str(count) + ' correlations processed'])
                print str(count) + ' correlations processed'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' correlations processed'])
    print str(count) + ' correlations processed'

def commandMicroarrayCoexpressionIntersectSIF(self, command):
    sqlstmt = "select distinct unigenel, unigenel2 from \
    pntscwlc intersect where unigenel!=unigenel2"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = open(str(command[1]), 'w')
    for clone in self.muscopedia.dbcursor:
        try: filename.write(clone[0].strip() + '\tpp\t' + clone[1].strip()
                            + os.linesep)
        except: pass
    filename.close()

def commandMicroarrayCoexpressionUnionSIF(self, command):
    sqlstmt = "select distinct unigenel, unigenel2 from all_coexp \
    where unigenel!=unigenel2"

```

```

self.muscopedia.dbcursor.execute(sqlstmt)
filename = open(str(command[1]), 'w')
for clone in self.muscopedia.dbcursor:
    try: filename.write(clone[0].strip() + '\tpp\t' + clone[1].strip()
        + os.linesep)
    except: pass
filename.close()

def commandMicroarrayLoad(self, command):
    """
    Invoked on 'maload' command. This command loads microarray data into
    the system. It takes the 'submitter' and 'description' to load into
    microarray_main table before processing the input file for loading into
    microarray_data_abs table, where the spot data will be stored. The input
    file format is {<clone ID, spot intensity>}, separated by '\t'.

    Usage: maload <submitter> <description> <file>
    """
    command = [command[0], ' '.join(command[1:-1]), command[-1]]
    self.muscopedia.dbcursor.execute('select gen_id(microarray_id_gen, 1)\
        from RDB$DATABASE')
    microarray_id = self.muscopedia.dbcursor.fetchall()[0][0]
    print microarray_id
    self.muscopedia.dbcursor.execute('insert into microarray_main \
        (microarrayid, description, submitter) values (?, ?, ?)',
        (microarray_id, command[2], command[1]))
    data = [(clone.split('\t')[0], clone.split('\t')[1][: -1])
        for clone in open(command[2], 'r').readlines()[: -1]]
    print 'Number of clones in ' + command[1] + ' file: ' + str(len(data))
    count = 0
    for clone in data:
        count = count + 1
        try:
            self.muscopedia.dbcursor.execute('insert into \
                microarray_data_abs (microarrayid, clone_id, intensity) \
                values (?, ?, ?)', (microarray_id, clone[0], clone[1]))
        except: pass
        if (count % 1000) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' clones processed for
                microarray_data_abs table'])
            print str(count) + ' clones processed for microarray_data_abs
                table'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' clones processed for microarray_data_abs
        table'])
    print str(count) + ' clones processed for microarray_data_abs table'

def muscopediaCycle(self):
    """Reconnect to Muscopedia."""
    try:
        self.muscopedia.dbconnect.commit()
        self.muscopedia.dbconnect.close() # shuts down Muscopedia Database
        command = 'localhost:' + os.path.join(self.environment["indir"],
            BioDatabases) + os.sep + 'muscopedia.fdb') + ':mouse:mouse'
        self.muscopedia = PubMedGrabber.FBUtilities(command)
    except: pass

def commandLinkPubMed(self):
    """
    Invoked on 'linkpubmed' command. It calls
    PubMedGrabber.PubMedSearch.__init__() to get connection to PubMed SOAP
    server.

    Usage: linkpubmed"""
    print "Trying to get SOAP link to PubMed Server..."
    self.result.append("Trying to get SOAP link to PubMed Server...")
    self.PostRun(self.result)
    del self.result[:]
    self.pubmed = PubMedGrabber.PubMedSearcher()
    self.result.append("PubMed linked.")

def commandPGCombination(self):
    """
    Invoked on 'pgcomb' command. It forms a list 2-element combinations
    from the names (entities) found for each PMID in pg_name table and loads
    it in pg_comb table.

```

```

Usage: pgcomb"""
from conceptmap import names
names = names.nameStatistics()
self.PostRun(['Obtaining PMIDs from pg_name table'])
print 'Obtaining PMIDs names from pg_name table'
self.muscopedia.dbcursor.execute('select pmid from pg_name')
pmidlist = names.uniqueNames([item[0] for item in
                             self.muscopedia.dbcursor.fetchall()])

pmid_count = 0
comb_count = 0
for pmid in pmidlist:
    self.muscopedia.dbcursor.execute('select distinct name from \
pg_name where pmid = '+ str(pmid))
    namelist = names.uniqueNames([item[0] for item in
                                self.muscopedia.dbcursor.fetchall()])
    combinations = UCFunctions.xcombinations(namelist, 2)
    for mix in combinations:
        self.muscopedia.dbcursor.execute('insert into pg_comb (pmid, \
namel, name2) values (?, ?, ?)', (int(pmid),mix[0],mix[1]))
        comb_count = comb_count + 1
    pmid_count = pmid_count + 1
    if (pmid_count % 500) == 0:
        self.muscopedia.dbconnect.commit()
        self.PostRun([str(comb_count) + ' name combinations in ' +
str(pmid_count) + ' abstracts inserted into pg_comb table'])
        print str(comb_count) + ' name combinations in ' +
str(pmid_count) + ' abstracts inserted into pg_comb table'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(comb_count) + ' name combinations in ' +
str(pmid_count) + ' abstracts inserted into pg_comb table'])
    print str(comb_count) + ' name combinations in ' + str(pmid_count) + '
abstracts inserted into pg_comb table'

def commandPGDistinctNames(self):
    """
    Invoked on 'pgdistinct' command. It gets a list of names (entities)
    from pg_name table and writes the list of distinct names into
    pg_distinct table.

    Usage: pgdistinct"""
    from conceptmap import names
    names = names.nameStatistics()
    self.PostRun(['Obtaining and processing names from pg_name table'])
    print 'Obtaining and processing names from pg_name table'
    self.muscopedia.dbcursor.execute('select name from pg_name')
    data = names.uniqueNames([item[0] for item in
                             self.muscopedia.dbcursor.fetchall()])
    self.PostRun([str(len(data)) + ' distinct names found'])
    print str(len(data)) + ' distinct names found'
    self.muscopedia.dbcursor.execute('delete from pg_distinct')
    self.muscopedia.dbconnect.commit()
    count = 0
    for name in data:
        try:
            self.muscopedia.dbcursor.execute('insert into pg_distinct \
(name) values (?)', (name,))
        except IOError: pass
        count = count + 1
        if (count % 10000) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' distinct names inserted into
pg_distinct table'])
            print str(count) + ' distinct names inserted into pg_distinct
table'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' distinct names inserted into pg_distinct
table'])
    print str(count) + ' distinct names inserted into pg_distinct table'

def commandPGGenBankID(self):
    """
    Invoked on 'pggi' command. It search for the GI from GenBank for each
    of the protein/gene names in pg_distinct table and load into pg_id
    table.

    Usage: pggi

```

```

"""
from Bio import GenBank
self.muscopedia.dbcursor.execute('select name from pg_distinct where \
    name not in (select name from pg_id)')
count = 0
for item in self.muscopedia.dbcursor.fetchall():
    try:
        IDList = GenBank.search_for(item[0])
        print IDList
        for id in IDList:
            try:
                print item[0]+' '+id
                self.muscopedia.dbcursor.execute('insert into pg_id \
                    (name, gi) values (?,?)', (item[0], id))
            except IndexError: print 'no match in GenBank for ' +
                str(item[0])
        except IndexError: print 'no match in GenBank for ' + str(item[0])
        count = count + 1
        if (count % 10) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' names processed for GenBank ID'])
            print str(count) + ' names processed for GenBank ID'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' names processed for GenBank ID'])
    print str(count) + ' names processed for GenBank ID'

def commandPGGenBankIDload(self, command):
    count = 0
    for item in marshal.load(open(os.path.join(self.environment["inidir"],
        command[1]))):
        try:
            self.muscopedia.dbcursor.execute('insert into pg_id \
                (name, gi) values (?,?)', (item[0], item[1]))
        except IOError: pass
        count = count + 1
        if (count % 1000) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' names processed for GenBank ID'])
            print str(count) + ' names processed for GenBank ID'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' names processed for GenBank ID'])
    print str(count) + ' names processed for GenBank ID'

def commandPGGenBankIDwrite(self, command):
    """
    Invoked on 'pggiwrite' command. It takes the file in command[1] as
    input file for processing with GenBank though BioPython and writes the
    results as [[name, gi]] using marshal module.

    Usage: pggiiwrite <file output from 'pggiup' command> <file to write
        to>
            pggiiwrite <file output from 'pggiup' command> <file to write
                to> <start number>
                    where both files must not include full path. The
                    files are in 'inidir'."""
    from Bio import GenBank
    pg = marshal.load(open(os.path.join(self.environment["inidir"],
        command[1])))
    data = []
    count = 0
    if len(command) == 4:
        print len(pg)
        pg = pg[int(command[3]):]
        print len(pg)
    for item in pg:
        try:
            IDList = GenBank.search_for(item[0])
            for id in IDList: data.append([item[0], id])
            count = count + 1
            self.PostRun([str(count) + ' of ' + str(len(pg)) + '
                processed'])
            print str(count) + ' of ' + str(len(pg)) + ' processed'
        except: pass
        if (count % 10) == 0:
            marshal.dump(data, open(os.path.join(
                self.environment["inidir"], command[2]), "w"))
            self.PostRun([str(count) + ' of ' + str(len(pg)) + ' wrote'])

```

```

        print str(count) + ' of ' + str(len(pg)) + ' wrote'
        marshal.dump(data, open(os.path.join(self.environment["inidir"],
        command[2]), "w"))

def commandPGGenBankIDunpro(self, command):
    """
    Invoked on 'pggiup' command. It writes the list of protein/gene names
    that are present in pg_distinct table but not in pg_id table into a file
    specified in command[1] using marshal module.

    Usage: pgguiup <output file>
           where <output file> must not include full path. The files are in
           'inidir'."""
    self.muscopedia.dbcursor.execute('select name from pg_distinct where \
        name not in (select name from pg_id)')
    marshal.dump(self.muscopedia.dbcursor.fetchall(),
        open(os.path.join(self.environment["inidir"], command[1]), "w"))

def commandPGHunt(self):
    """
    Invoked on 'pghunt' command. It retrieves abstract from pmc_abstract
    table one at a time using the PMID list in name_hunt_waiting table. Each
    abstract is processed by GAPSCORE at Stanford University and only
    results with score higher than 0.95 are inserted into the pg_name table.
    After each abstract is processed, its PMID will be deleted from
    name_hunt_waiting table.

    Reference: Jeffrey T. Chang, Hinrich Schutze, Russ B. Altman. 2004.
        GAPSCORE: finding gene and protein names one word at a time.
        Bioinformatics 20(2): 216-225.

    Usage: pghunt"""
    from abcrawl import bionlp
    self.PostRun(["Obtaining PMID list from name_hunt_waiting table"])
    print "Obtaining PMID list from name_hunt_waiting table"
    self.muscopedia.dbcursor.execute('select pmid from name_hunt_waiting')
    data = self.muscopedia.dbcursor.fetchall()
    self.PostRun(["Data from name_hunt_waiting table extracted"])
    print "Data from name_hunt_waiting table extracted"
    print str(len(data)) + " abstract(s) to process"
    # data is [(

```



```

        processed."
        if (abstract_number % 5000) == 0:
            self.muscopedia.dbconnect.commit()
        self.muscopedia.dbconnect.commit()

def commandProcessCloneSoundex(self):
    """
    Invoked on 'pclonesoundex' command. It generates the soundex
    representation of each clone's description and abbreviated name and
    cache it in clone_soundex table.

    Usage: pclonesoundex
    """
    self.muscopedia.dbcursor.execute('select cloneid, otherid, idtype, \
                                     description, abb_desc from clone_view')

    clone = []
    for c in self.muscopedia.dbcursor.fetchall():
        t = []
        for ele in c:
            try: t.append(ele.strip())
            except AttributeError: t.append('')
        clone.append(t)
    clone = [[c[0], c[1], c[2], UCFFunctions.soundex(c[3], 20),
              UCFFunctions.soundex(c[4], 5)] for c in clone]

    count = 0
    self.muscopedia.dbcursor.execute('delete from clone_soundex')
    self.muscopedia.dbconnect.commit()
    for c in clone:
        stmt = "insert into clone_soundex (cloneid, otherid, idtype, \
            desc_sound, abb_sound) values ('%s','%s','%s','%s','%s') " \
            % (str(c[0]), str(c[1]), str(c[2]), str(c[3]), str(c[4]))
        try:
            print stmt
            self.muscopedia.dbcursor.execute(stmt)
        except TypeError: pass
        count = count + 1
        if (count % 500) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' clones processed for soundex'])
            print str(count) + ' clones processed for soundex'
    self.PostRun([str(count) + ' clones processed for soundex'])
    print str(count) + ' clones processed for soundex'

def commandPoissonSIF(self, command):
    sqlstmt = "select p.clone1, p.clone2 from jclone_occur_pair p \
              inner join jclone_occur_stat s on p.clone1=s.clone1 and \
              p.clone2=s.clone2"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = str(command[1])
    clone = [[clone[0].strip(), clone[1].strip()]
              for clone in self.muscopedia.dbcursor.fetchall()]
    filename = open(filename, 'w')
    for c in clone: filename.write(c[0] + '\tpp\t' + c[1] + os.linesep)
    filename.close()

def commandPoissonEDA(self, command):
    sqlstmt = "select p.clone1, p.clone2, p.paircount, s.poisson99 from \
              jclone_occur_pair p inner join jclone_occur_stat s on \
              p.clone1=s.clone1 and p.clone2=s.clone2"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = str(command[1])
    clone = [[clone[0].strip(), clone[1].strip(), clone[2], clone[3]]
              for clone in self.muscopedia.dbcursor.fetchall()]
    filename = open(filename, 'w')
    for c in clone: filename.write(c[0] + '\t(pp)\t' + c[1] + ' = ' +
                                   str(c[2] - c[3]) + os.linesep)
    filename.close()

def commandPubGeneOneSIF(self, command):
    sqlstmt = "select unigenel, unigene2 from pubgene_as_unigene where \
              unigenel!=unigene2 and unigene2!=unigenel"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = str(command[1])
    clone = [[clone[0].strip(), clone[1].strip()] for clone in
              self.muscopedia.dbcursor.fetchall()]
    filename = open(filename, 'w')
    for c in clone: filename.write(c[0] + '\tpp\t' + c[1] + os.linesep)

```

```

filename.close()

def commandPubGeneFiveSIF(self, command):
    sqlstmt = "select unigenel, unigene2 from pubgene_as_unigene where \
        paircount > 4 and unigenel!=unigene2 and unigene2!=unigenel"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = str(command[1])
    clone = [[clone[0].strip(), clone[1].strip()] for clone in
        self.muscopedia.dbcursor.fetchall()]
    filename = open(filename, 'w')
    for c in clone: filename.write(c[0] + '\tpp\t' + c[1] + os.linesep)
    filename.close()

def commandPubGeneOneVerb(self):
    self.muscopedia.dbcursor.execute('delete from pgl_verb')
    self.muscopedia.dbconnect.commit()
    sqlstmt = "select distinct unigenel, unigene2 from \
        pubgene_as_unigene and unigenel!=unigene2"
    self.muscopedia.dbcursor.execute(sqlstmt)
    clone = [[clone[0].strip(), clone[1].strip()] for clone in
        self.muscopedia.dbcursor.fetchall()]
    count = 0
    for c in clone:
        sqlstmt = "select verb from ml_svo where subject containing '%s' \
            and object containing '%s'" % (c[0], c[1])
        self.muscopedia.dbcursor.execute(sqlstmt)
        verb = [v.strip() for v in self.muscopedia.dbcursor.fetchall()]
        if len(verb) == 0: verb.append('no_interact')
        for v in verb:
            sqlstmt = "insert into pgl_verb (subject, verb, object) \
                values (?, ?, ?)"
            try:
                self.muscopedia.dbcursor.execute(sqlstmt, (c[0], v, c[1]))
            except: pass
            count = count + 1
            if (count % 100) == 0:
                self.muscopedia.dbconnect.commit()
                self.PostRun([str(count) + ' interacting pairs processed'
                    for ' + table])
                print str(count) + ' interacting pairs processed for ' +
                    table
        self.muscopedia.dbconnect.commit()
        self.PostRun([str(table) + ' table processed'])
        print str(table) + ' table processed'

def commandPGPubMedIndex(self, command):
    """
    Invoked on 'pgindex' command.

    Usage: pgindex
           pgindex <reverse>
    """
    self.muscopedia.dbcursor.execute('select jclone from
job_clone_master')
    count = 0
    clist = [clone[0].strip() for clone in
self.muscopedia.dbcursor.fetchall()]
    if len(command) > 1 and str(command[1]) == 'reverse':
        clist.reverse()
    for clone in clist:
        self.commandPubMedIndexUpdate(['pmuindex', str(clone)])
        count = count + 1
        self.result.append(str(count) + ' clone processed.')
        print str(count) + ' clone processed.'

def commandPNcoexpLoad(self, command):
    f = open(command[1], 'r')
    self.muscopedia.dbcursor.execute("delete from pn_coexp")
    self.muscopedia.dbconnect.commit()
    count = 0
    for line in f:
        line = line.split('\t')
        self.muscopedia.dbcursor.execute("insert into pn_coexp (clone1, \
            clone2, correlation) values ('%s', '%s', '%3.3f')" \
            % (line[1], line[2], float(line[3][:-1])))
        count = count + 1
        if (count % 10000) == 0:

```

```

        self.muscopedia.dbconnect.commit()
        self.PostRun([str(count) + ' correlations processed'])
        print str(count) + ' correlations processed'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' correlations processed'])
    print str(count) + ' correlations processed'

def commandPNcoexpSIF(self, command):
    sqlstmt = "select unigenel, unigene2 from pn_coexp_as_unigene where \
    correlation > 0.75 or correlation < -0.75"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = open(str(command[1]), 'w')
    for clone in self.muscopedia.dbcursor:
        try: filename.write(clone[0].strip() + '\tpp\t' +
            clone[1].strip() + os.linesep)
        except: pass
    filename.close()

def commandPopulateGalactopoesisDifferential(self):
    self.muscopedia.dbcursor.execute('delete from galactopoesis_up')
    self.muscopedia.dbcursor.execute('delete from galactopoesis_down')
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.execute("insert into galactopoesis_up select \
        distinct(unigene) from ts_array_data where \
        ((Lac3_C+lac3_a+lac3_b)/3)/((Lac1_C+lac1_a+ \
        lac1_b)/3) > 2 or lac3_a/lac1_a > 2 or lac3_b/lac1_b > 2")
    self.muscopedia.dbcursor.execute("insert into galactopoesis_down \
        select distinct(unigene) from ts_array_data where \
        ((Lac3_C+lac3_a+lac3_b)/3)/((Lac1_C+lac1_a+ lac1_b)/3) < \
        0.5 or lac3_a/lac1_a < 0.5 or lac3_b/lac1_b < 0.5")
    self.muscopedia.dbconnect.commit()

def commandPopulateLactogenesisDifferential(self):
    self.muscopedia.dbcursor.execute('delete from lactogenesis_up')
    self.muscopedia.dbcursor.execute('delete from lactogenesis_down')
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.execute("insert into lactogenesis_up select \
        unigene from ts_array_data where \
        ((Lac1_C+lac1_a+lac1_b)/3)/((P17_5_C+p17_5_a+p17_5_b)/3)>\
        2 or lac1_c/p17_5_c > 2")
    self.muscopedia.dbcursor.execute("insert into lactogenesis_up select \
        a.unigene from pn_array_data p join affy_annot a on \
        p.probeid=a.probeid where (((p.NFD1Lact1+p.NFD1Lact2+ \
        p.NFD1Lact3+p.NFD1Lact4)/4)/ \
        ((p.NFD17Preg1+p.NFD17Preg2+p.NFD17Preg3+p.NFD17Preg4)/4)) \
        > 2) or (p.NFD1Lact1 / p.NFD17Preg1 > 2) or \
        (p.NFD1Lact2 / p.NFD17Preg2 > 2) or \
        (p.NFD1Lact3 / p.NFD17Preg3 > 2) or \
        (p.NFD1Lact4 / p.NFD17Preg4 > 2)")
    self.muscopedia.dbcursor.execute("insert into lactogenesis_down \
        select unigene from ts_array_data where \
        ((Lac1_C+lac1_a+lac1_b)/3)/((P17_5_C+p17_5_a+ \
        p17_5_b)/3) < 0.5 or lac1_c/p17_5_c < 0.5")
    self.muscopedia.dbcursor.execute("insert into lactogenesis_down \
        select a.unigene from pn_array_data p join affy_annot a \
        on p.probeid=a.probeid where (((p.NFD1Lact1+p.NFD1Lact2+ \
        p.NFD1Lact3+p.NFD1Lact4)/4)/((p.NFD17Preg1+p.NFD17Preg2+ \
        p.NFD17Preg3+p.NFD17Preg4)/4) < 0.5) or (p.NFD1Lact1 / \
        p.NFD17Preg1 < 0.5) or (p.NFD1Lact2 / p.NFD17Preg2 < 0.5) \
        or (p.NFD1Lact3 / p.NFD17Preg3 < 0.5) or \
        (p.NFD1Lact4 / p.NFD17Preg4 < 0.5)")
    self.muscopedia.dbconnect.commit()

def commandPopulateIFPDifferential(self):
    self.muscopedia.dbcursor.execute('delete from IFPdiff')
    # upregulated in IFP
    sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
        sifp10/sfp10 > 1.2 and sifp10/sip10 > 1.2 and sifp10/sif10 > 1.2"
    self.muscopedia.dbcursor.execute(sqlstmt)
    unigene = [('uIFP', x[0].strip()) for x in
        self.muscopedia.dbcursor.fetchall()]
    self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
        unigene) values (?,?)", unigene)

    # upregulated in IF
    sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
        (sifp10/sfp10 > 1.2) and (sifp10/sip10 > 1.2) and not \
        (sifp10/sif10 > 1.2)"

```

```

self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('uIF-', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
                                     unigene) values (?,?)", unigene)
# upregulated in IP
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
          (sifp10/sfp10 > 1.2) and (sifp10/sif10 > 1.2) and not \
          (sifp10/sip10 > 1.2)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('uI-P', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
                                     unigene) values (?,?)", unigene)
# upregulated in FP
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
          (sifp10/sfp10 > 1.2) and (sifp10/sip10 > 1.2) and not \
          (sifp10/sfp10 > 1.2)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('u-FP', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
                                     unigene) values (?,?)", unigene)
# upregulated in I
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
          (sifp10/sfp10 > 1.2) and not (sifp10/sip10 > 1.2) and not \
          (sifp10/sif10 > 1.2)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('uI--', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
                                     unigene) values (?,?)", unigene)
# upregulated in F
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
          (sifp10/sip10 > 1.2) and not (sifp10/sfp10 > 1.2) and not \
          (sifp10/sif10 > 1.2)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('u-F-', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
                                     unigene) values (?,?)", unigene)
# upregulated in P
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
          (sifp10/sif10 > 1.2) and not (sifp10/sip10 > 1.2) and not \
          (sifp10/sfp10 > 1.2)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('u--P', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
                                     unigene) values (?,?)", unigene)
# downregulated in IFP
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
          sifp10/sfp10 < 0.8 and sifp10/sip10 < 0.8 and sifp10/sif10 < 0.8"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('dIFP', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
                                     unigene) values (?,?)", unigene)
# downregulated in IF
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
          (sifp10/sfp10 < 0.8) and (sifp10/sip10 < 0.8) and not \
          (sifp10/sif10 < 0.8)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('dIF-', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
                                     unigene) values (?,?)", unigene)
# downregulated in IP
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
          (sifp10/sfp10 < 0.8) and (sifp10/sif10 < 0.8) and not \
          (sifp10/sip10 < 0.8)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('dI-P', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
                                     unigene) values (?,?)", unigene)
# downregulated in FP

```

```

sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
(sifp10/sif10 < 0.8) and (sifp10/sip10 < 0.8) and not \
(sifp10/sfp10 < 0.8)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('d-FP', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
unigene) values (?,?)", unigene)
# downregulated in I
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
(sifp10/sfp10 < 0.8) and not (sifp10/sip10 < 0.8) and not \
(sifp10/sif10 < 0.8)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('dI--', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
unigene) values (?,?)", unigene)
# downregulated in F
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
(sifp10/sip10 < 0.8) and not (sifp10/sfp10 < 0.8) and not \
(sifp10/sif10 < 0.8)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('d-F-', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
unigene) values (?,?)", unigene)
# downregulated in P
sqlstmt = "select distinct unigene from km_mouse_vitro_id where \
(sifp10/sif10 < 0.8) and not (sifp10/sip10 < 0.8) and not \
(sifp10/sfp10 < 0.8)"
self.muscopedia.dbcursor.execute(sqlstmt)
unigene = [('d--P', x[0].strip()) for x in
            self.muscopedia.dbcursor.fetchall()]
self.muscopedia.dbcursor.executemany("insert into IFPdiff (diff, \
unigene) values (?,?)", unigene)

def commandProcessOccurrence(self, command):
    """
    Invoked on 'processoccurrence' command.

    Usage: processoccurrence
    """
    self.muscopedia.dbcursor.execute('delete from jclone_occur_pair')
    self.muscopedia.dbcursor.execute('delete from jclone_occur_single')
    self.muscopedia.dbconnect.commit()
    self.muscopedia.dbcursor.execute('select jclone from \
job_clone_master')
    clone = [clone[0].strip() for clone in
              self.muscopedia.dbcursor.fetchall()]
    print "List of clones: " + str(clone)
    count = 0
    # processing single clones
    for c in clone:
        stmt = "insert into jclone_occur_single (clone, paircount) \
select '%s', count(pmid) from jclone_occurrence_pmid \
where clone = '%s' % (c, c)"
        try: self.muscopedia.dbcursor.execute(stmt)
        except: pass
        count = count + 1
        if (count % 100) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' clones inserted into \
jclone_occur_single table'])
            print str(count) + ' clones inserted into jclone_occur_single \
table'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' clones inserted into jclone_occur_single \
table'])
    print str(count) + ' clones inserted into jclone_occur_single table'
    # processing pairwise occurrence
    count = 0
    for c in clone:
        for d in clone:
            stmt = "insert into jclone_occur_pair (clone1, clone2, \
paircount) select '%s', '%s', count(jcop1.pmid) from \
jclone_occurrence_pmid jcop1 where jcop1.clone = '%s' \
and exists (select 1 from jclone_occurrence_pmid jcop2 \

```

```

        where jcop2.pmid = jcop1.pmid and jcop2.clone = '%s'%" % \
        (c, d, c, d)
    try: self.muscopedia.dbcursor.execute(stmt)
    except: pass
    count = count + 1
    if (count % 100) == 0:
        self.muscopedia.dbconnect.commit()
        self.PostRun([str(count) + ' clone-pairs inserted into
            jclone_occur_pair table'])
        print str(count) + ' clone-pairs inserted into
            jclone_occur_pair table'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' clone-pairs inserted into
        jclone_occur_pair table'])
    print str(count) + ' clone-pairs inserted into jclone_occur_pair
        table'

def commandPubMedBootStrapFetch(self, command):
    """
    Invoked by 'pmbsfetch' command. This method primarily does the same as
    commandPubMedFetch() method. It differs by fetching a record and loading
    into pmc_* tables one at a time. Removes the use of
    commandPubMedLoadDB() method. Suitable for use with
    commandPubMedReturnMax() and commandPubMedReturnStart() methods to
    download large numbers of records

    Usage: pmbsfetch"""
    try:
        # retrieving PMIDs from PubMedSearcher.runSearch()
        self.result.append("Fetching data using the term(s): " +
            str(command[1]))
        print "Fetching data using the term(s): " + str(command[1])
        self.PostRun(self.result)
        del self.result[:]
        self.pubmed.runSearch(str(command[1]))
        self.result.append(str(len(self.pubmed.resultSet.IDList)) + "
            records found")
        print str(len(self.pubmed.resultSet.IDList)) + " records found"
        # retrieve and process each article
        article_count = 0
        total_article_count = len(self.pubmed.resultSet.IDList)
        for ID in self.pubmed.resultSet.IDList:
            gc.collect()
            try:
                if (self.muscopedia._checkPMID(ID, 'pmc_article_info') ==
                    'TRUE'):
                    # article present in database
                    # check presence of search term in the database
                    self.PostRun(['PMID: ' + str(ID) + ' found in
                        database. Proceed to update index'])
                    print 'PMID: ' + str(ID) + ' found in database.
                        Proceed to update index'
                    searchTerm = string.split(str(command[1]), '+')
                    for term in searchTerm:
                        try:
                            if (self.muscopedia._checkTerm(term,
                                'pmc_index_list') == 'FALSE'):
                                # term not in database
                                self.muscopedia.dbcursor.execute('insert \
                                    into pmc_index_list (term, indextype) \
                                    values (?, ?)',(str(term), \
                                    'all_fields'))
                                self.muscopedia.sqlstmt = "select termid \
                                    from pmc_index_list where term = '%s'\"
                                    % term
                                self.muscopedia.dbcursor.execute(
                                    self.muscopedia.sqlstmt)
                                termid =
                                    self.muscopedia.dbcursor.fetchone()
                                self.muscopedia.dbcursor.execute('insert \
                                    into pmc_index_term_map (pmid, termid)\
                                    values (?, ?)', (ID, termid[0]))
                            else: # term in pmc_term_list
                                # link between article and index term
                                self.muscopedia.sqlstmt = "select termid \
                                    from pmc_index_list where term = '%s'\"

```

```

        % term
        self.muscopedia.dbcursor.execute(
            self.muscopedia.sqlstmt)
        termid =
            self.muscopedia.dbcursor.fetchone()
        self.sqlstmt = 'select pmid from \
            pmc_index_term_map where termid = %s \
            and pmid = %s' % (termid[0], ID)
        self.muscopedia.dbcursor.execute(
            self.muscopedia.sqlstmt)
        if len(self.muscopedia.dbcursor.fetchall()
            ) == 0: #no link
            self.muscopedia.dbcursor.execute(
                'insert into pmc_index_term_map\
                (pmid, termid) values (?, ?)',
                (ID, termid[0]))
        self.PostRun(['PMID: ' + str(ID) + ' index
            update complete.'])
        print 'PMID: ' + str(ID) + ' index update
            complete.'
    except:
        self.PostRun(['PMID: ' + str(ID) + '
            processing failed.'])
        print 'PMID: ' + str(ID) + ' processing
            failed.'
        # added to give more information about failure
        self.result = list(formatExceptionInfo())
        self.PostRun(self.result)
        article_count = article_count + 1
        self.PostRun([str(article_count) + ' of ' +
            str(total_article_count) + ' completed.'])
        print str(article_count) + ' of ' +
            str(total_article_count) + ' completed.'
    else:
        # if article not in database, proceed to download article
        try:
            article = self.pubmed.runFetchOne(ID)
            # load article into primary tables
            self.muscopedia.directLoadDB(str(command[1]),
                article)
            self.PostRun(['PMID: ' + str(ID) + ' fetched and
                committed to primary tables'])
            print 'PMID: ' + str(ID) + ' fetched and committed
                to primary tables'
        except:
            self.PostRun(["Error for PMID abstract: " +
                str(data[count][0])])
            print "Error for PMID abstract: " +
                str(data[count][0])
            article_count = article_count + 1
            self.PostRun([str(article_count) + ' of ' +
                str(total_article_count) + ' completed.'])
            print str(article_count) + ' of ' +
                str(total_article_count) + ' completed.'
    except:
        self.PostRun(['PMID: ' + str(ID) + ' processing failed.'])
        print 'PMID: ' + str(ID) + ' processing failed.'
        # added to give more information about failure
        self.result = list(formatExceptionInfo())
        self.PostRun(self.result)
        article_count = article_count + 1
        if (article_count % 100) == 0:
            self.muscopedia.dbconnect.commit()
        self.muscopedia.dbconnect.commit()
    except: self.result.append("Command malformed. Type 'help pmfetch' for
        help on 'pmfetch' command.")
    self.muscopedia.dbconnect.commit()

def commandPubMedFetch(self, command):
    """
    Invoked on 'pmfetch' command. It calls for PubMedGrabber.
    PubMedSearcher.runFetch() to download article information into
    PubMedGrabber.PubMedSearcher.resultSet structure using command[1] as the
    search term.

    Usage: pmfetch <search term>
    pmfetch -a <search term>"""

```

```

print "Searching PubMed and fetching data....."
gc.collect()
try:
    self.result.append("Fetching data using the term(s): " +
        str(command[1]))
    print "Fetching data using the term(s): " + str(command[1])
    self.PostRun(self.result)
    del self.result[:]
    self.pubmed.runSearch(str(command[1]))
    self.result.append(str(len(self.pubmed.resultSet.IDList)) + "
        records found")
    print str(len(self.pubmed.resultSet.IDList)) + " records found"
    self.pubmed.runFetch (str(command[1]))
    self.result.append("Fetch data completed.")
    self.result.append("Records fetched: " +
        str(len(self.pubmed.resultSet.IDList)))
except: self.result.append("Command malformed. Type 'help pmfetch' for
    help on 'pmfetch' command.")

def commandPubMedFetchPro(self, command):
    """
    Invoked by 'pmfetchpro' command. This method primarily does the same
    as commandPubMedFetch() method but is a more sophisticated version that
    is recommended for normal use. Instead of simply loading new citations
    into the pmc_* tables, this method also calls for abbreviation
    processing and load the abbreviated abstract into text_replace_abstract
    table and text_replace_abstract_upro table. The latter insertion is to
    prepare for 'svoosnew' command.
    PMID also inserted into name_hunt_waiting table in preparation for
    'pghunt' command.

    Usage: pmfetchpro"""
    try:
        # retrieving PMIDs from PubMedSearcher.runSearch()
        self.result.append("Fetching data using the term(s): " +
            str(command[1]))
        print "Fetching data using the term(s): " + str(command[1])
        self.PostRun(self.result)
        del self.result[:]
        self.pubmed.runSearch(str(command[1]))
        self.result.append(str(len(self.pubmed.resultSet.IDList)) + "
            records found")
        print str(len(self.pubmed.resultSet.IDList)) + " records found"
        # preparing text replacer system
        from abcrawl import TextReplacer
        TRRuleDir = os.path.join(self.environment["inidir"], "abcrawl" +
            os.sep)
        TRRuleSet = []
        for rulefile in RRFile:
            TRRuleSet.append(TRRuleDir + rulefile)
        TR = TextReplacer.ReplacerSet(TRRuleSet)
        self.PostRun(["Replacement rule file(s) used: " + str(TRRuleSet),
            "Text Replacer ready"])
        print "Replacement rule file(s) used: " + str(TRRuleSet)
        print "Text Replacer ready"
        # retrieve and process each article
        article_count = 0
        total_article_count = len(self.pubmed.resultSet.IDList)
        for ID in self.pubmed.resultSet.IDList:
            gc.collect()
            try:
                if (self.muscopedia._checkPMID(ID, 'pmc_article_info') ==
                    'TRUE'):
                    # article present in database
                    # check presence of search term in the database
                    self.PostRun(['PMID: ' + str(ID) + ' found in
                        database. Proceed to update index'])
                    print 'PMID: ' + str(ID) + ' found in database.
                        Proceed to update index'
                    searchTerm = string.split(str(command[1]), '+')
                    for term in searchTerm:
                        try:
                            if (self.muscopedia._checkTerm(term,
                                'pmc_index_list') == 'FALSE'):
                                # term not in database
                                self.muscopedia.dbcursor.execute('insert \
                                    into pmc_index_list (term, indextype)\

```



```

        values (?, ?)', (str(term),
        'all_fields'))
self.muscopedia.sqlstmt = "select termid \
from pmc_index_list where term = '%s'\"
% term
self.muscopedia.dbcursor.execute(
    self.muscopedia.sqlstmt)
termid =
    self.muscopedia.dbcursor.fetchone()
self.muscopedia.dbcursor.execute('insert \
into pmc_index_term_map (pmid, termid)\
values (?, ?)', (ID, termid[0]))
else: # term in pmc_term_list
    # link between article and index term
    self.muscopedia.sqlstmt = "select termid \
from pmc_index_list where term = '%s'\"
% term
self.muscopedia.dbcursor.execute(
    self.muscopedia.sqlstmt)
termid =
    self.muscopedia.dbcursor.fetchone()
self.sqlstmt = 'select pmid from \
pmc_index_term_map where termid = %s \
and pmid = %s' % (termid[0], ID)
self.muscopedia.dbcursor.execute(
    self.muscopedia.sqlstmt)
if len(self.muscopedia.dbcursor.fetchall()
) == 0:
    #no link
    self.muscopedia.dbcursor.execute(
        'insert into pmc_index_term_map\
(pmid, termid) values (?, ?)',\
(ID, termid[0]))
self.PostRun(['PMID: ' + str(ID) + ' index
update complete.'])
print 'PMID: ' + str(ID) + ' index update
complete.'
except:
    self.PostRun(['PMID: ' + str(ID) + '
processing failed.'])
print 'PMID: ' + str(ID) + ' processing
failed.'
# added to give more information about failure
self.result = list(formatExceptionInfo())
self.PostRun(self.result)
article_count = article_count + 1
self.PostRun([str(article_count) + ' of ' +
str(total_article_count) + ' completed.'])
print str(article_count) + ' of ' +
str(total_article_count) + ' completed.'
else:
    # if article not in database, proceed to download article
    article = self.pubmed.runFetchOne(ID)
    # load article into primary tables
    self.muscopedia.directLoadDB(str(command[1]), article)
    self.PostRun(['PMID: ' + str(ID) + ' fetched and
committed to primary tables'])
    print 'PMID: ' + str(ID) + ' fetched and committed to
primary tables'
    # process abstract for word substitution
    try:
        text =
            TR.substituteMultiple(article.abstractList[0])
        self.muscopedia.dbcursor.execute("insert into \
text_replace_abstract (pmid, text) values \
(?,?)", (ID, text))
    # insert into processing backlog,
    text_replace_abstract_upro table
        self.muscopedia.dbcursor.execute("insert into \
text_replace_abstract_upro (pmid, text)
values (?,?)", (ID, text))
    # added for protein/gene name processing
    stmt = "insert into name_hunt_waiting (pmid) \
values (" + ID + ")"
    self.muscopedia.dbcursor.execute(stmt)
    self.PostRun(["PMID " + str(ID) + " abstract
processed and committed to Muscopedia"])

```

```

        print "PMID " + str(ID) + " abstract processed and
            committed to Muscopedia"
    except:
        self.PostRun(["Error for PMID abstract: " +
            str(data[count][0])])
        print "Error for PMID abstract: " +
            str(data[count][0])
        article_count = article_count + 1
        self.PostRun([str(article_count) + ' of ' +
            str(total_article_count) + ' completed.'])
        print str(article_count) + ' of ' +
            str(total_article_count) + ' completed.'
    except:
        self.PostRun(['PMID: ' + str(ID) + ' processing failed.'])
        print 'PMID: ' + str(ID) + ' processing failed.'
        # added to give more information about failure
        self.result = list(formatExceptionInfo())
        self.PostRun(self.result)
        article_count = article_count + 1
    if (article_count % 100) == 0:
        self.muscopedia.dbconnect.commit()
        self.PostRun(['Cycle Muscopedia'])
        print 'Cycle Muscopedia'
        self.muscopediaCycle()
        self.muscopedia.dbconnect.commit()
    except: self.result.append("Command malformed. Type 'help pmfetch' for
        help on 'pmfetch' command.")
    self.muscopedia.dbconnect.commit()

def commandPubMedLoadDB(self):
    """
    Invoked on 'pmloaddb' command. It loads the last fetched results from
    PubMed (fetched using pmfetch command) into Muscopedia Database, which
    is linked up using 'linkmuscopedia' command.

    Usage: pmloaddb"""
    self.result.append("Loading PubMed Results into Muscopedia
        Database.....")
    print "Loading PubMed Results into Muscopedia Database....."
    self.PostRun(self.result)
    del self.result[:]
    self.muscopedia.directLoadDB(self.pubmed.history[-1],
        self.pubmed.resultSet)
    self.result.append("Loading completed")

def commandPubMedIndexUpdate(self, command):
    """
    Invoked on 'pmuindex' command. It retrieves a list of PMIDs with the
    keywords (search terms) and update Muscopedia Database. Only PMIDs that
    are already present in the database will be updated.

    Usage: pmuindex <search term>
    """
    print "Searching PubMed and fetching data....."
    gc.collect()
    try:
        self.result.append("Fetching data using the term(s): " +
            str(command[1]))
        print "Fetching data using the term(s): " + str(command[1])
        self.PostRun(self.result)
        del self.result[:]
        self.pubmed.runSearch(str(command[1]))
        self.result.append(str(len(self.pubmed.resultSet.IDList)) + "
            records found")
        self.muscopedia.updateIndexList(self.pubmed.history[-1],
            self.pubmed.resultSet)
    except: pass
    self.result.append("Loading completed")

def commandPubMedReturnMax(self, command):
    """
    Invoked on 'pmreturnlimit' command. It can either prints out the
    current PubMed fetch return limit (number of articles' information to
    download) or change the limit.

    Usage: pmreturnlimit
        pmreturnlimit <new limit>"""

```

```

        if (len(command) < 2):
            self.result.append('PubMed return limit: ' +
                               str(self.pubmed.return_max))
        else:
            self.result.append('PubMed return limit: ' +
                               str(self.pubmed.return_max))
            self.pubmed.return_max = int(command[1])
            self.result.append('PubMed return limit changed')
            self.result.append('New PubMed return limit: ' +
                               str(self.pubmed.return_max))

def commandPubMedReturnStart(self, command):
    """
    Invoked on 'pmreturnstart' command. It can either prints out the
    current PubMed fetch return start point or change the start point. If
    '100' means future fetches will only start from 100th from the latest
    record.

    Usage: pmreturnstart
           pmreturnstart <new start>"""
    if (len(command) < 2):
        self.result.append('PubMed return start: ' +
                           str(self.pubmed.return_start))
    else:
        self.result.append('PubMed return start: ' +
                           str(self.pubmed.return_start))
        self.pubmed.return_start = int(command[1])
        self.result.append('PubMed return start changed')
        self.result.append('New PubMed return start: ' +
                           str(self.pubmed.return_start))

def commandPubMedWriteFile(self, command):
    """
    Invoked on 'pmwritefile' command. It writes the last search result
    (using pmfetch command) to file. The name of the file is given in
    command[1]. If absolute file name is given, the file will be written to
    the absolute file path. If absolute file name is not given, then the
    file will be written into the current working directory.

    Usage: pmwritefile <file name> """
    if os.path.isabs(str(command[1])):
        self.pubmed.writeResultsToFile(str(command[1]), 'w')
    else:
        self.pubmed.writeResultsToFile(os.path.join(
            self.environment["cwd"], str(command[1])), 'w')

def commandQuit(self):
    """
    Invoked on 'quit' command. It forms a path to the
    'mosirium_environment.ini' file using the 'inidir' environment variable
    (which should point to the same directory as this application is
    located), opens the file and write the environment variables
    (self.environment) into the file, using marshal module. It finally
    exits the application using sys.exit() command."""
    try:
        self.muscopedia.dbconnect.commit()
        self.muscopedia.dbconnect.close() # shuts down Muscopedia Database
    except: pass
    f = open(os.path.join(self.environment["inidir"],
                          "muscorian_environment.ini"),
             "w")
    marshal.dump(self.environment, f)
    f.close()
    sys.exit()

def commandRMEC(self, command):
    """
    Invoked on 'rmec' command.

    Usage: rmec <entity>+
    """
    import math
    data = {}
    data['clonename'] = ['N', 'mean', 'variance', 'maximum', 'minimum',
                        'stddev', 'rmec']
    data['no_data'] = []
    for clone in command[1:]:
        sqlstmt = "select count(microarray_data_abs.intensity), \"

```

```

        avg(microarray_data_abs.intensity), \
        (sum(microarray_data_abs.intensity* \
            microarray_data_abs.intensity)/ \
            (count(microarray_data_abs.intensity)+1))- \
        (((sum(microarray_data_abs.intensity)* \
            sum(microarray_data_abs.intensity))/ \
            ((count(microarray_data_abs.intensity)+1)* \
            count(microarray_data_abs.intensity))))), \
        max(microarray_data_abs.intensity), \
        min(microarray_data_abs.intensity) \
        from microarray_clone_map inner join microarray_data_abs on \
        (microarray_clone_map.clone_id=microarray_data_abs.clone_id) \
        inner join job_clone_map on (microarray_clone_map.other_id=\
        job_clone_map.other_id) where job_clone_map.jclone='%s' " % \
        clone.upper())
self.muscopedia.dbcursor.execute(sqlstmt)
temp = list(self.muscopedia.dbcursor.fetchall()[0])
try:
    temp.append(math.sqrt(temp[2]))
    # calculate standard deviation from variance
    temp.append(temp[1]/temp[5]) # calculate RMEC
    data[clone] = temp
except: data['no_data'] = data['no_data'].append(clone)
self.PostRun(["Data for " + clone + " obtained"])
print "Data for " + clone + " obtained"
print data

```

```

def commandScanNewAbbreviations(self):
    """
    Invoked on 'scannewabb' command. It uses BioNLP web service at
    Stanford University to scan 20000 abstracts in text_replace_abstract
    table for new abbreviations and write new abbreviations (more than 0.9
    in score) into ReplacementRule.MDF file.

    Reference: Chang JT, Schutze H, and Altman RB. Creating an Online
    Dictionary of Abbreviations from MEDLINE. The Journal of the
    American Medical Informatics Association 9(6): 612-20, 2002."""
    from abcrawl import bionlp
    from abcrawl import TextReplacer
    gc.collect()
    self.PostRun(["Extracting data from text_replace_abstract table"])
    print "Extracting data from text_replace_abstract table"
    self.muscopedia.dbcursor.execute('select pmid, text from \
        pmc_abstract')
    data = self.muscopedia.dbcursor.fetchall()
    self.PostRun(["Data from text_replace_abstract table extracted"])
    print "Data from text_replace_abstract table extracted"
    print str(len(data)) + " abstract(s) to process"
    abblast = []
    if len(data) > 20005: del data[19999:]
    # take 20k abstracts to process if more than 20005 in all
    for record in data: abblast = abblast +
        bionlp.find_abbreviations(record[1])
    self.PostRun([str(len(data)) + " abstract(s) processed"])
    print str(len(data)) + " abstract(s) processed"
    self.PostRun([str(len(abblast)) + " abbreviations found"])
    print str(len(abblast)) + " new abbreviations found"
    f = open(os.path.join(self.environment["inidir"], "abcrawl" + os.sep +
        "ReplacementRule.MDF"), 'a')
    count = 0
    for abbreviation in abblast:
        if abbreviation[2] > 0.88:
            f.write(str(abbreviation[0]) + "=" + str(abbreviation[1]) +
                os.linesep)
            count = count + 1
    self.PostRun([str(count) + " out of " + str(len(abblast)) + "
        abbreviations found had score above cutoff (0.88)"])
    print str(count) + " out of " + str(len(abblast)) + " abbreviations
        found had score above cutoff (0.88)"
    self.PostRun([str(count) + " abbreviations written into "+
        os.path.join(self.environment["inidir"], "abcrawl" + os.sep +
        "ReplacementRule.MDF")])
    print str(count) + " abbreviations written into "+
        os.path.join(self.environment["inidir"], "abcrawl" + os.sep +
        "ReplacementRule.MDF")
    f.close()

```

```

self.PostRun([os.path.join(self.environment["inidir"], "abcrawl" +
    os.sep + " ReplacementRule.MDF") + "file closed."])
print os.path.join(self.environment["inidir"], "abcrawl" + os.sep +
    "ReplacementRule.MDF") + " file closed."
TR = TextReplacer.Replacer(' ', os.path.join(
    self.environment["inidir"], "abcrawl" + os.sep +
    "ReplacementRule.MDF"))
TR.cleanDictionaryFile(os.path.join(self.environment["inidir"],
    "abcrawl" + os.sep + "ReplacementRule.MDF"))
self.PostRun([os.path.join(self.environment["inidir"], "abcrawl" +
    os.sep + "ReplacementRule.MDF") + " file cleaned"])
print os.path.join(self.environment["inidir"], "abcrawl" + os.sep +
    "ReplacementRule.MDF") + " file cleaned"

def commandSVOOSAll(self):
    """
    Invoked on 'svoosall' command. It cleans out ml_svo table and use
    MontyLingua
    (by Hugo Liu, MIT media labs) to process every abstract in
    text_replace_abstract
    table for subject-verb-object (SVO) structures.
    """
    self.PostRun(["Extracting data from text_replace_abstract table"])
    print "Extracting data from text_replace_abstract table"
    self.muscopedia.dbcursor.execute('select pmid from
text_replace_abstract')
    data = self.muscopedia.dbcursor.fetchall()
    self.PostRun(["Data from text_replace_abstract table extracted"])
    print "Data from text_replace_abstract table extracted"
    print str(len(data)) + " abstract(s) to process"
    gc.collect()
    svooslist = []
    pmid = 0
    total_record = len(data)
    record_count = 0
    for record in data:
        pmid = record[0]
        self.muscopedia.dbcursor.execute('select text from \
            text_replace_abstract where pmid = ' + str(pmid))
        text = self.muscopedia.dbcursor.fetchall()
        svooslist = UCFunctions.ML_svoosProcess(
            self.montylingua.jlist_predicates(text[0][0]))
        for svoos in svooslist:
            if len(svoos) < 3: svoos.append(' ')
            try: self.muscopedia.dbcursor.execute("insert into ml_svo \
                (pmid, verb, subject, object) values (?, ?, ?, ?)", \
                (pmid, svoos[0], svoos[1], svoos[2]))
            except: pass
        print "SVOOS from PMID " + str(pmid) + " committed."
        self.PostRun(["SVOOS from PMID " + str(pmid) + " committed."])
        record_count = record_count + 1
        self.PostRun([str(record_count) + ' of ' + str(total_record) + '
            completed.'])
        print str(record_count) + ' of ' + str(total_record) + '
            completed.'
        if (record_count % 5000) == 0: self.muscopedia.dbconnect.commit()
    self.muscopedia.dbconnect.commit()
    print str(len(data)) + " abstract(s) processed by MontyLingua"
    self.PostRun([str(len(data)) + " abstract(s) processed by
        MontyLingua"])
    self.muscopedia.dbcursor.execute("select count(*) from ml_svo")
    pmid = self.muscopedia.dbcursor.fetchall()
    # pmid now holds number of records in ml_svo table
    print str(pmid[0][0]) + " subject-verb-object tuple entered in
        Muscopedia database"
    self.PostRun([str(pmid[0][0]) + " subject-verb-object tuples entered
        in Muscopedia database"])

def commandSVOOSNew(self):
    """
    Invoked on 'svoosnew' command. It compares the PMIDs of two tables in
    Muscopedia, text_replace_abstract_upro (which contains substituted but
    unprocessed abstracts inserted by commandPubMedFetchPro() method) and
    ml_svo (which contains the SVOs), to obtain a list of PMIDs that are
    found in text_replace_abstract but not in ml_svo. Using the PMIDs,
    abstracts are extracted and processed using MontyLingua (by Hugo Liu,
    MIT media labs) to obtain subject-verb-object structures."""

```

```

self.PostRun(["Extracting data from text_replace_abstract_upro
table"])
print "Extracting data from text_replace_abstract_upro table"
self.muscopedia.dbcursor.execute('select pmid, text from \
text_replace_abstract_upro')
data = self.muscopedia.dbcursor.fetchall()
self.PostRun(["Data from text_replace_abstract_upro table extracted"])
print "Data from text_replace_abstract_upro table extracted"
print str(len(data)) + " abstract(s) to process"
gc.collect()
svooslist = []
pmid = 0
total_record = len(data)
record_count = 0
for record in data:
    pmid = record[0]
    svooslist = UCFunctions.ML_svoosProcess(
        self.montylingua.jist_predicates(record[1]))
    for svoos in svooslist:
        if len(svoos) < 3: svoos.append(' ')
        try: self.muscopedia.dbcursor.execute("insert into ml_svo \
(pmid, verb, subject, object) values (?, ?, ?, ?)", \
(pmid, svoos[0], svoos[1], svoos[2]))
        except: pass
    print "SVOOS from PMID " + str(pmid) + " committed."
    self.PostRun(["SVOOS from PMID " + str(pmid) + " committed."])
    record_count = record_count + 1
    # delete record in text_replace_abstract_upro table
    try:
        self.muscopedia.dbcursor.execute("delete from \
text_replace_abstract_upro where pmid = " + str(pmid))
        self.PostRun(['PMID: ' + str(pmid) + ' deleted from \
text_replace_abstract_upro'])
        print 'PMID: ' + str(pmid) + ' deleted from \
text_replace_abstract_upro'
    except: pass
    self.PostRun([str(record_count) + ' of ' + str(total_record) + '
completed.'])
    print str(record_count) + ' of ' + str(total_record) + '
completed.'
    if (record_count % 5000) == 0: self.muscopedia.dbconnect.commit()
self.muscopedia.dbconnect.commit()
print str(len(data)) + " abstract(s) processed by MontyLingua"
self.PostRun([str(len(data)) + " abstract(s) processed by
MontyLingua"])
self.muscopedia.dbcursor.execute("select count(*) from ml_svo")
pmid = self.muscopedia.dbcursor.fetchall()
# pmid now holds number of records in ml_svo table
print str(pmid[0][0]) + " subject-verb-object tuple entered in
Muscopedia database"
self.PostRun([str(pmid[0][0]) + " subject-verb-object tuples entered
in Muscopedia database"])

def commandTextReplaceAll(self):
    """
    Invoked on 'textreplaceall' command. It clears the
    text_replace_abstract table in Muscopedia Database before processing all
    the abstracts from pmc_abstract table using TextReplacer module.
    Processed abstracts are written into text_replace_abstract table."""
    from abcrawl import TextReplacer
    TRRuleDir = os.path.join(self.environment["inidir"], "abcrawl" +
os.sep)
    TRRuleSet = []
    for rulefile in RRFile:
        TRRuleSet.append(TRRuleDir + rulefile)
    TR = TextReplacer.ReplacerSet(TRRuleSet)
    self.PostRun(["Replacement rule file(s) used: " + str(TRRuleSet),
"Text Replacer ready"])
    print "Replacement rule file(s) used: " + str(TRRuleSet)
    print "Text Replacer ready"
    self.muscopedia.dbcursor.execute('delete from text_replace_abstract')
    self.PostRun(["text_replace_abstract table cleared"])
    print "text_replace_abstract table cleared"
    self.PostRun(["Extracting data from pmc_abstract table"])
    print "Extracting data from pmc_abstract table"
    self.muscopedia.dbcursor.execute('select pmid, text from \

```

```

                                pmc_abstract')
data = self.muscopedia.dbcursor.fetchall()
self.PostRun(["Data from pmc_abstract table extracted"])
print "Data from pmc_abstract table extracted"
print str(len(data)) + " abstract(s) to process"
gc.collect()
total_record = len(data)
record_count = 0
for count in range(len(data)):
    try:
        text = TR.substituteMultiple(data[count][1])
        self.muscopedia.dbcursor.execute("insert into \
            text_replace_abstract (pmid, text) values (?,?)", \
            (data[count][0], text))
# double entry into text_replace_abstract_upro is removed b'cos trying to
# insert and delete entries at the same time slows down the whole system
        self.PostRun(["PMID " + str(data[count][0]) + " abstract
            processed and committed to Muscopedia"])
        print "PMID " + str(data[count][0]) + " abstract processed and
            committed to Muscopedia"
    except:
        self.PostRun(["Error for PMID abstract: " +
            str(data[count][0])])
        print "Error for PMID abstract: " + str(data[count][0])
        result = list(formatExceptionInfo())
        record_count = record_count + 1
        self.PostRun([str(record_count) + ' of ' + str(total_record) + '
            completed.'])
        print str(record_count) + ' of ' + str(total_record) + '
            completed.'
        if (record_count % 5000) == 0: self.muscopedia.dbconnect.commit()
self.muscopedia.dbconnect.commit()
self.PostRun([str(record_count) + " abstract(s) processed"])
print str(len(record_count)) + " abstract(s) processed"

def commandTScoexpLoad(self, command):
    f = open(command[1], 'r')
    self.muscopedia.dbcursor.execute("delete from ts_coexp_as_unigene")
    self.muscopedia.dbconnect.commit()
    count = 0
    for line in f:
        line = line.split('\t')
        self.muscopedia.dbcursor.execute("insert into ts_coexp_as_unigene\
            (unigene1, unigene2, correlation) values ('%s', '%s', '%3.3f')" \
            % (line[1], line[2], float(line[3][:-1])))
        count = count + 1
        if (count % 10000) == 0:
            self.muscopedia.dbconnect.commit()
            self.PostRun([str(count) + ' correlations processed'])
            print str(count) + ' correlations processed'
    self.muscopedia.dbconnect.commit()
    self.PostRun([str(count) + ' correlations processed'])
    print str(count) + ' correlations processed'

def commandTScoexpSIF(self, command):
    sqlstmt = "select unigene1, unigene2 from ts_coexp_as_unigene where \
        correlation > 0.75 or correlation < -0.75"
    self.muscopedia.dbcursor.execute(sqlstmt)
    filename = open(str(command[1]), 'w')
    for clone in self.muscopedia.dbcursor:
        try: filename.write(clone[0].strip() + '\tpp\t' + clone[1].strip()
            + os.linesep)
        except: pass
    filename.close()

def commandPrototype(self, command):
    """
    Method prototype for command processor methods."""
    if (command[1] == "<something>"):
        try:
            # do work here
            return
        except: self.result.append("Command malformed. Type 'help
            <prototype>' for help on 'prototype' command.")
    else: self.result.append("<prototype> " + str(command[1]) + " command
        unrecognized.")

```

```

# ----- End of Command Processors -----

def run(self, command):
    """
    Method to sort the commands to the respective command processors.
    Takes a list of commands (each word of the command is an element in the
    list, with the 1st element in the list being the actual word of command)
    as sole variable, route it to the respective command processor method,
    and returns the results as a list."""
    self.result = []
    if (command[0] == "copyright"): self.result = ExtendedCopyright()
    elif (command[0] == 'abstractsort'): self.commandAbstractSort()
    elif (command[0] == 'abstract80'): self.commandAbstract80()
    elif (command[0] == 'activateeda'): self.commandActivateEDA(command)
    elif (command[0] == 'activatesif'): self.commandActivateSIF(command)
    elif (command[0] == 'asactivate0'): self.commandASactivate0()
    elif (command[0] == 'asactivate80'): self.commandASactivate80()
    elif (command[0] == 'asbind0'): self.commandASbind0()
    elif (command[0] == 'asbind80'): self.commandASbind80()
    elif (command[0] == 'asdegrade0'): self.commandASdegrade0()
    elif (command[0] == 'asdegrade80'): self.commandASdegrade80()
    elif (command[0] == 'asenhance0'): self.commandASenhance0()
    elif (command[0] == 'asenhance80'): self.commandASenhance80()
    elif (command[0] == 'asinhbit0'): self.commandASinhbit0()
    elif (command[0] == 'asinhbit80'): self.commandASinhbit80()
    elif (command[0] == 'asppi50'): self.commandASppi50()
    elif (command[0] == 'asactivate_d'): self.commandASactivateDistinct()
    elif (command[0] == 'asbind_d'): self.commandASbindDistinct()
    elif (command[0] == 'bindeda'): self.commandBindEDA(command)
    elif (command[0] == 'bindsif'): self.commandBindSIF(command)
    elif (command[0] == 'cooccurrenceeda'):
        self.commandCooccurrenceEDA(command)
    elif (command[0] == 'cwcoexpressionload'):
        self.commandCWcoexpLoad(command)
    elif (command[0] == 'cwcoexpressionsif'):
        self.commandCWcoexpSIF(command)
    elif (command[0] == 'degradeeda'): self.commandDegradeEDA(command)
    elif (command[0] == 'degradesif'): self.commandDegradeSIF(command)
    elif (command[0] == 'dispro'): self.commandDiseaseProtein(command)
    elif (command[0] == 'dm_2entityinteract'):
        self.commandDM_2entityinteraction(command)
    elif (command[0] == 'enhanceeda'): self.commandEnhanceEDA(command)
    elif (command[0] == 'enhancesif'): self.commandEnhanceSIF(command)
    elif (command[0] == 'genmacorintersect'):
        self.commandGenerateMicroarrayCorrelationIntersects()
    elif (command[0] == 'genmacorunion'):
        self.commandGenerateMicroarrayCorrelationUnion()
    elif (command[0] == 'genmaexpstats'):
        self.commandGenerateMicroarrayExpStatistics()
    elif (command[0] == 'gencache'):
        self.commandGenerateNonMicroarrayCache()
    elif (command[0] == 'inhibitededa'): self.commandInhibitEDA(command)
    elif (command[0] == 'inhibitsif'): self.commandInhibitSIF(command)
    elif (command[0] == 'kmglucocorticoid1'):
        self.commandKMglucocorticoid1()
    elif (command[0] == 'kmglucocorticoid1out'):
        self.commandKMglucocorticoid1out(command)
    elif (command[0] == 'kminsul1'): self.commandKminsul1()
    elif (command[0] == 'kminsul1out'):
        self.commandKminsul1out(command)
    elif (command[0] == 'kmprolactin1'): self.commandKmprolactin1()
    elif (command[0] == 'kmprolactin1out'):
        self.commandKmprolactin1out(command)
    elif (command[0] == 'jcloneoccur'): self.commandJobCloneOccurrence()
    elif (command[0] == 'jcloneoccurind'):
        self.commandJobCloneOccurrenceIndividual()
    elif (command[0] == 'jcloneoccurpoisson'):
        self.commandJobCloneOccurrencePoisson()
    elif (command[0] == 'jobcloneload'): self.commandJobCloneLoad(command)
    elif (command[0] == 'lccoexpressionload'):
        self.commandLCcoexpLoad(command)
    elif (command[0] == 'lccoexpressionsif'):
        self.commandLCcoexpSIF(command)
    elif (command[0] == 'linkall'): self.commandLinkAll(command)
    elif (command[0] == 'linkmontylingua'): self.commandLinkMontyLingua()
    elif (command[0] == 'linkmuscopedia'):

```



```

        self.commandLinkMuscopediaDB(command)
    elif (command[0] == 'linkpubmed'): self.commandLinkPubMed()
    elif (command[0] == 'loadaccugll'): self.commandLoadAccUGLL(command)
    elif (command[0] == 'loadaffyannt'): self.commandLoadAffyAnnt(command)
    elif (command[0] == 'loadkmdata'): self.commandLoadKMdata(command)
    elif (command[0] == 'loadpndata'): self.commandLoadPNdata(command)
    elif (command[0] == 'loadtsdata'): self.commandLoadTSdata(command)
    elif (command[0] == 'makenamestatmap'):
        self.commandMakePGNameMap(command)
    elif (command[0] == 'makebindingmap'):
        self.commandMakeBindingMap(command)
    elif (command[0] == 'makebindingmapsif'):
        self.commandMakeBindingMapSIF(command)
    elif (command[0] == 'makebindingmapfromlistexact'):
        self.commandMakeBindingMapFromListExact(command)
    elif (command[0] == 'makeactivationmap'):
        self.commandMakeActivationMap(command)
    elif (command[0] == 'makeactivationmapsif'):
        self.commandMakeActivationMapSIF(command)
    elif (command[0] == 'macorrelation'):
        self.commandMicroarrayCorrelation()
    elif (command[0] == 'macorrelationintersectsif'):
        self.commandMicroarrayCoexpressionIntersectSIF(command)
    elif (command[0] == 'macorrelationunionsif'):
        self.commandMicroarrayCoexpressionUnionSIF(command)
    elif (command[0] == 'maload'): self.commandMicroarrayLoad(command)
    elif (command[0] == 'macloneload'):
        self.commandMicroarrayClonesLoad(command)
    elif (command[0] == 'macloneidload'):
        self.commandMicroarrayClonesIDLoad(command)
    elif (command[0] == 'pclonesoundex'):
        self.commandProcessCloneSoundex()
    elif (command[0] == 'pgcomb'): self.commandPGCombination()
    elif (command[0] == 'pgdistinct'): self.commandPGDistinctNames()
    elif (command[0] == 'pggi'): self.commandPGGenBankID()
    elif (command[0] == 'pggiload'): self.commandPGGenBankIDload(command)
    elif (command[0] == 'pggiup'): self.commandPGGenBankIDunpro(command)
    elif (command[0] == 'pggiwrite'):
        self.commandPGGenBankIDwrite(command)
    elif (command[0] == 'pgindex'): self.commandPGPubMedIndex(command)
    elif (command[0] == 'pghunt'): self.commandPGHunt()
    elif (command[0] == 'pmbsfetch'):
        self.commandPubMedBootStrapFetch(command)
    elif (command[0] == 'pmfetch'): self.commandPubMedFetch(command)
    elif (command[0] == 'pmfetchpro'): self.commandPubMedFetchPro(command)
    elif (command[0] == 'pmloaddb'): self.commandPubMedLoadDB()
    elif (command[0] == 'pmreturnlimit'):
        self.commandPubMedReturnMax(command)
    elif (command[0] == 'pmreturnstart'):
        self.commandPubMedReturnStart(command)
    elif (command[0] == 'pmuindex'):
        self.commandPubMedIndexUpdate(command)
    elif (command[0] == 'pmwritefile'):
        self.commandPubMedWriteFile(command)
    elif (command[0] == 'pncoexpressionload'):
        self.commandPNcoexpLoad(command)
    elif (command[0] == 'pncoexpressionsif'):
        self.commandPNcoexpSIF(command)
    elif (command[0] == 'poissoneda'): self.commandPoissonEDA(command)
    elif (command[0] == 'poissonsif'): self.commandPoissonSIF(command)
    elif (command[0] == 'populatelactogenesisdiff'):
        self.commandPopulateLactogenesisDifferential()
    elif (command[0] == 'populategalactopoesisdiff'):
        self.commandPopulateGalactopoesisDifferential()
    elif (command[0] == 'populateifpdiff'):
        self.commandPopulateIFPDifferential()
    elif (command[0] == 'processoccurrence'):
        self.commandProcessOccurrence(command)
    elif (command[0] == 'pubgeneonesif'):
        self.commandPubGeneOneSIF(command)
    elif (command[0] == 'pubgenefivesif'):
        self.commandPubGeneFiveSIF(command)
    elif (command[0] == 'pubgeneoneverb'): self.commandPubGeneOneVerb()
    elif (command[0] == 'quit'): self.commandQuit()
    elif (command[0] == 'rmec'): self.commandRMEC(command)
    elif (command[0] == 'scannewabb'): self.commandScanNewAbbreviations()
    elif (command[0] == 'svoosall'): self.commandSVOOSAll()

```

```

        elif (command[0] == 'svoosnew'): self.commandSVOOSNew()
        elif (command[0] == 'textreplaceall'): self.commandTextReplaceAll()
        elif (command[0] == 'tscoexpressionload'):
            self.commandTScoexpLoad(command)
        elif (command[0] == 'tscoexpressionsif'):
            self.commandTScoexpSIF(command)
        else: self.UnknownCommandError()
        return self.result

def formatExceptionInfo(maxTBlevel=10):
    """
    Method to gather information about an exception raised. It is used to
    readout the exception messages and type of exception. This method takes a
    parameter, maxTBlevel, which is set to 10, which defines the maximum level
    of tracebacks to recall.

    This method is obtained from
    http://www.linuxjournal.com/article.php?sid=5821"""
    cla, exc, trbk = sys.exc_info()
    excName = cla.__name__
    try:
        excArgs = exc.__dict__["args"]
    except KeyError:
        excArgs = "<no args>"
    excTb = traceback.format_tb(trbk, maxTBlevel)
    return (excName, excArgs, excTb)

def main():
    command = ""
    mosys = MuscorianSystem()
    mosys.environment_initialize()
    Copyright()
    while 1:
        try:
            command = ""
            command = raw_input("Muscorian:> ")
            command = string.split(command, " ")
            mosys.PreRun(command)
            result = mosys.run(command)
            mosys.PostRun(result)
        except: result = list(formatExceptionInfo())
        if (command[0] == "quit"):
            mosys.commandQuit()
        for line in result:
            if (type(line) == list):
                for l in line:
                    print l
            print line

if __name__ == '__main__':
    global command
    main()

```

## Path Name: muscorian/UCFunctions.py

```

def ML_svoosExpand(c_svoos):
    svoos = string.split(c_svoos, '|')
    data = []
    if len(svoos) > 3:
        data.append(svoos[0] + '|' + svoos[1] + '|' + ' '.join(svoos[2:]))
    else: data.append(''.join(svoos))
    return data

def ML_svoos_cleanup(text):
    seg = re.compile(r'"\'s"')
    brv = re.compile(r'(\(|\)|\'|\")')
    text = seg.sub(r'|', text)
    return brv.sub(r'', text)

def ML_svoosProcess(svoos):
    data = []
    temp = []
    for block in svoos:
        for line in block:
            temp = temp + list(ML_svoosExpand(ML_svoos_cleanup(str(line))))

```

```

data = [string.split(line, '|')
        for line in temp
        if len(string.split(line, '|')) > 2:
            and not string.split(line, '|')[1] == '']
return data

def xcombinations(items, n):
    if n==0: yield []
    else:
        for i in xrange(len(items)):
            for cc in xcombinations(items[:i]+items[i+1:],n-1):
                yield [items[i]]+cc

def sample_wr(population, k):
    import random
    n = len(population)
    _random, _int = random.random, int # speed hack
    result = [None] * k
    for i in xrange(k):
        j = _int(_random() * n)
        result[i] = population[j]
    return result

def soundex(name, len=4):
    digits = '01230120022455012623010202'
    sndx = ''
    fc = ''
    for c in name.upper():
        if c.isalpha():
            if not fc: fc = c # remember first letter
            d = digits[ord(c)-ord('A')]
            # duplicate consecutive soundex digits are skipped
            if not sndx or (d != sndx[-1]):
                sndx += d
    sndx = fc + sndx[1:]
    sndx = sndx.replace('0','')
    return (sndx + (len * '0'))[:len]

def set_from_list(t):
    clean = {}
    for combination in t: clean[combination] = 1
    return clean.keys()

```

## Path Name: muscorian/abcrawl/TextReplacer.py

```

class Replacer(UserDict):
    def __init__(self, dict = None, file = None):
        self.re = None
        self.regex = None
        if file == None:
            UserDict.__init__(self, dict)
            self._make_regex()
        else:
            UserDict.__init__(self, self.readDictionaryFile(file))
            self._make_regex()
    def cleanDictionaryFile(self, file):
        import os
        dict = self.readDictionaryFile(file)
        f = open(file, 'w')
        for key in dict.keys(): f.write(str(key) + '=' + str(dict[key]) +
            os.linesep)
        f.close()
    def readDictionaryFile(self, file):
        import string
        import os
        f = open(file, 'r')
        data = f.readlines()
        f.close()
        dict = {}
        for rule in data:
            rule = rule.split('=')
            if rule[1][-1] == os.linesep: rule[1] = rule[1][:-1]
            dict[str(rule[0])] = str(rule[1])
        print '%s replacement rule(s) read from %s' % (str(len(dict.keys())),
            str(file))

```

```

        return dict
    def _make_regex(self):
        self.re = "(%s)" % "|".join(map(re.escape, self.keys()))
        self.regex = re.compile(self.re)
    def __call__(self, mo):
        self.count += 1 # Look-up string
        return self[mo.string[mo.start():mo.end()]]
    def substitute(self, text):
        self.count = 0
        return self.regex.sub(self, text)
    def rmBracketDuplicate(self, text):
        regex = re.compile(r'(\w+)\s*(\(\1\))')
        return regex.sub(r'\1', text)
    def substituteMultiple(self, text):
        count = 1 # to get into the loop
        run = 0 # counter for number of runs thru the text
        while count > 0 and run < 10:
            count = 0
            text = self.rmBracketDuplicate(self.substitute(text))
            count = count + self.count
            run = run + 1
            print "Pass %d: Changed %d things(s)" % (run, count)
        return text

class ReplacerSet:
    regexSet = []
    def __init__(self, filelist):
        for file in filelist: self.regexSet.append(Replacer('', file))
    def rmBracketDuplicate(self, text):
        regex = re.compile(r'(\w+)\s*(\(\1\))')
        return regex.sub(r'\1', text)
    def substitute(self, text):
        self.count = 0
        for regex in self.regexSet:
            text = regex.substitute(text)
            self.count = self.count + regex.count
        return text
    def substituteMultiple(self, text):
        count = 1 # to get into the loop
        run = 0 # counter for number of runs thru the text
        while count > 0 and run < 10:
            count = 0
            text = self.rmBracketDuplicate(self.substitute(text))
            count = count + self.count
            run = run + 1
            print "Pass %d: Changed %d things(s)" % (run, count)
        return text

```

## Path Name: muscorian/abcrawl/PubMedGrabber.py

```

class PubMedResultSet:
    resultDictionary = {}
    IDList = []
    count = 0
    abstractList = []
    titleList = []
    authorList = []
    ISSNList = []
    issueList = []
    volumeList = []
    pageList = []
    DOIList = []
    def clearall( self ):
        self.resultDictionary = {}
        self.abstractList = []
        self.titleList = []
        self.count = 0
        self.authorList = []
        self.ISSNList = []
        self.issueList = []
        self.volumeList = []
        self.pageList = []
        self.DOIList = []
    def processIDList( self ):
        index = 0
        for id in self.IDList:

```

```

        self.resultDictionary[str( id )] = index
        index = index + 1

class PubMedConnector:
    from SOAPpy import WSDL
    self.pbserv = WSDL.Proxy( wsdl )

class PubMedSearcher:
    history = []
    return_max = 100000
    return_start = 1
    serv = ''
    db = 'pubmed'
    resultSet = PubMedResultSet( )
    def __init__( self, wsdl =
        'http://eutils.ncbi.nlm.nih.gov/entrez/eutils/soap/v1.1/eutils.wsdl' ):
        self.serv = PubMedConnector( wsdl ).pbserv
    def searchConstructor( self, str ):
        c_str = ''
        strList = string.split( str, ' ' )
        c_str = strList.pop( 0 )
        if ( len( strList ) > 0 ):
            for s in strList: c_str = c_str + '+' + s
        else: c_str = str
        self.history.append( c_str )
    def runSearch( self, term ):
        self.resultSet.clearall( )
        self.searchConstructor( term )
        result = self.serv.run_eSearch( db = self.db, term = self.history[-1],
            retmax = self.return_max, retstart = self.return_start )
        if len(result) > 0:
            self.resultSet.IDList = result.IdList['Id']
            self.resultSet.processIDList( )
            self.resultSet.count = len( self.resultSet.IDList )
        else: self.resultSet.count = 0
    def runFetch( self, term = '' ):
        if ( ( term == '' ) and (self.resultSet.count > 0 ) ):
            for ID in self.resultSet.IDList:
                result = self.serv.run_eFetch( db = self.db, id = ID )
                try:
                    self.resultSet.titleList.append(
                        result['PubmedArticle']['MedlineCitation']
                        ['Article']['ArticleTitle'] )
                except AttributeError: self.resultSet.titleList.append('')
                try:
                    self.resultSet.authorList.append(
                        result['PubmedArticle']['MedlineCitation']
                        ['Article']['AuthorList'] )
                except AttributeError: self.resultSet.authorList.append('')
                try:
                    self.resultSet.abstractList.append(
                        result['PubmedArticle']['MedlineCitation']
                        ['Article']['Abstract']['AbstractText'] )
                except AttributeError: self.resultSet.abstractList.append('')
                try:
                    self.resultSet.ISSNList.append(
                        result['PubmedArticle']['MedlineCitation']
                        ['Article']['Journal']['ISSN'] )
                except AttributeError: self.resultSet.ISSNList.append('')
                try:
                    self.resultSet.volumeList.append(
                        result['PubmedArticle']['MedlineCitation']
                        ['Article']['Journal']['JournalIssue']['Volume'] )
                except AttributeError: self.resultSet.volumeList.append('')
                try:
                    self.resultSet.pageList.append(
                        result['PubmedArticle']['MedlineCitation']
                        ['Article']['Pagination']['MedlinePgn'] )
                except AttributeError: self.resultSet.pageList.append('')
                try:
                    self.resultSet.DOIList.append(result['PubmedArticle']
                        ['PubmedData']['ArticleIdList']['ArticleId'][1] )
                except AttributeError: self.resultSet.DOIList.append('')
            else:
                self.runSearch( term )
                if self.resultSet.count > 0:
                    for ID in self.resultSet.IDList:

```

```

        result = self.serv.run_eFetch( db = self.db, id = ID )
        try:
            self.resultSet.titleList.append(
                result['PubmedArticle']['MedlineCitation']
                ['Article']['ArticleTitle'] )
        except AttributeError: self.resultSet.titleList.append('')
        try:
            self.resultSet.authorList.append(
                result['PubmedArticle']['MedlineCitation']
                ['Article']['AuthorList'] )
        except AttributeError:
            self.resultSet.authorList.append('')
        try:
            self.resultSet.abstractList.append(
                result['PubmedArticle']['MedlineCitation']
                ['Article']['Abstract']['AbstractText'] )
        except AttributeError:
            self.resultSet.abstractList.append('')
        try:
            self.resultSet.ISSNList.append(
                result['PubmedArticle']
                ['Article']['Journal']['ISSN'] )
        except AttributeError: self.resultSet.ISSNList.append('')
        try:
            self.resultSet.volumeList.append(
                result['PubmedArticle']['MedlineCitation']
                ['Article']['Journal']['JournalIssue']['Volume'] )
        except AttributeError:
            self.resultSet.volumeList.append('')
        try:
            self.resultSet.pageList.append(
                result['PubmedArticle']['MedlineCitation']
                ['Article']['Pagination']['MedlinePgn'] )
        except AttributeError: self.resultSet.pageList.append('')
        try:
            self.resultSet.DOIList.append(
                result['PubmedArticle']['PubmedData']
                ['ArticleIdList']['ArticleId'][1] )
        except AttributeError: self.resultSet.DOIList.append('')
def runFetchOne(self, ID):
    result = PubMedResultSet()
    result.IDList.append(ID)
    r = self.serv.run_eFetch( db = self.db, id = ID )
    try:
        result.titleList.append( r['PubmedArticle']['MedlineCitation']
                                ['Article']['ArticleTitle'] )
    except AttributeError: self.resultSet.titleList.append('')
    try:
        result.authorList.append( r['PubmedArticle']['MedlineCitation']
                                ['Article']['AuthorList'] )
    except AttributeError: self.resultSet.authorList.append('')
    try:
        result.abstractList.append( r['PubmedArticle']['MedlineCitation']
                                ['Article']['Abstract']['AbstractText'] )
    except AttributeError: self.resultSet.abstractList.append('')
    try:
        result.ISSNList.append(r['PubmedArticle']['MedlineCitation']
                                ['Article']['Journal']['ISSN'] )
    except AttributeError: self.resultSet.ISSNList.append('')
    try:
        result.volumeList.append( r['PubmedArticle']['MedlineCitation']
                                ['Article']['Journal']['JournalIssue']['Volume'] )
    except AttributeError: self.resultSet.volumeList.append('')
    try:
        result.pageList.append( r['PubmedArticle']['MedlineCitation']
                                ['Article']['Pagination']['MedlinePgn'] )
    except AttributeError: self.resultSet.pageList.append('')
    try:
        result.DOIList.append( r['PubmedArticle']['PubmedData']
                                ['ArticleIdList']['ArticleId'][1] )
    except AttributeError: self.resultSet.DOIList.append('')
    return result

class FBUtilities:
    dbhost = ''
    dbfile = ''

```

```

dbuser = ''
dbpwd = ''
dbconnect = ''
dbcursor = ''
sqlstmt = ''
def __init__( self, fbfullpath ):
    fbfullpath = string.split( fbfullpath, ':' )
    self.dbhost = string.strip( fbfullpath[0] )
    self.dbfile = string.strip( fbfullpath[1] )
    self.dbuser = string.strip( fbfullpath[2] )
    self.dbpwd = string.strip( fbfullpath[3] )
    if self.dbhost == 'localhost': self.dbhost = None
    self.dbconnect = kinterbasdb.connect( host = self.dbhost, database =
        self.dbfile, user = self.dbuser, password = self.dbpwd,
        charset='UNICODE_FSS')
    self.dbcursor = self.dbconnect.cursor( )
def close( self ):
    self.dbconnect.commit( )
    self.dbcursor.close( )
    self.dbconnect.close( )
def cleanDB( self ):
    self.dbcursor.execute('delete from pmc_article_info')
    self.dbcursor.execute('delete from pmc_abstract')
    self.dbcursor.execute('delete from pmc_author')
    self.dbcursor.execute('delete from pmc_index_list')
    self.dbcursor.execute('delete from pmc_index_term_map')
    self.dbconnect.commit( )
def _checkPMID(self, pmid, table):
    self.sqlstmt = 'select count(*) from ' +table+ ' where pmid = ' + pmid
    self.dbcursor.execute( self.sqlstmt )
    if self.dbcursor.fetchall( )[0][0] > 0: return 'TRUE'
    else: return 'FALSE'
def _checkTerm(self, term, table):
    self.sqlstmt = "select count(*) from %s where term = '%s'" %
        (table, term)
    self.dbcursor.execute( self.sqlstmt )
    if self.dbcursor.fetchall( )[0][0] > 0: return 'TRUE'
    else: return 'FALSE'
def updateIndexList(self, searchTerm, dataSet):
    searchTerm = string.split(searchTerm, '+')
    record_count = 0
    for index in range( len(dataSet.IDList) ):
        record_count = record_count + 1
        if (self._checkPMID(dataSet.IDList[index], 'pmc_article_info') ==
            'TRUE'):
            for term in searchTerm:
                try:
                    if (self._checkTerm(term, 'pmc_index_list') ==
                        'FALSE'):
                        self.dbcursor.execute('insert into pmc_index_list\
                            (term, indextype) values (?, ?)', \
                            (str(term), 'all_fields'))
                        self.sqlstmt = "select termid from pmc_index_list\
                            where term = '%s'" % term
                        self.dbcursor.execute(self.sqlstmt)
                        termid = self.dbcursor.fetchone()
                        self.dbcursor.execute('insert into \
                            pmc_index_term_map (pmid, termid) values \
                            (?, ?)', (dataSet.IDList[index], termid[0]))
                except:
                    self.sqlstmt = "select termid from pmc_index_list\
                        where term = '%s'" % term
                    self.dbcursor.execute(self.sqlstmt)
                    termid = self.dbcursor.fetchone()
                    self.sqlstmt = 'select pmid from \
                        pmc_index_term_map where termid = %s and \
                        pmid = %s' % (termid[0], \
                        dataSet.IDList[index])
                    self.dbcursor.execute(self.sqlstmt)
                    if len(self.dbcursor.fetchall( )) == 0: #no link
                        self.dbcursor.execute('insert into \
                            pmc_index_term_map (pmid, termid) values \
                            (?, ?)', (dataSet.IDList[index], termid[0]))
                except: pass
            if (record_count % 500) == 0: self.dbconnect.commit()
    self.dbconnect.commit()
def directLoadDB( self, searchTerm, dataSet ):

```

```

searchTerm = string.split(searchTerm, '+')
process_count = 0
for index in range( len(dataSet.IDList) ):
    process_count = process_count + 1
    if (self._checkPMID(dataSet.IDList[index], 'pmc_article_info') ==
        'TRUE'):
        for term in searchTerm:
            try:
                if (self._checkTerm(term, 'pmc_index_list') ==
                    'FALSE'):
                    self.dbcursor.execute('insert into pmc_index_list\
                        (term, indextype) values (?, ?)', \
                        (str(term), 'all_fields'))
                    self.sqlstmt = "select termid from pmc_index_list\
                        where term = '%s'" % term
                    self.dbcursor.execute(self.sqlstmt)
                    termid = self.dbcursor.fetchone()
                    self.dbcursor.execute('insert into \
                        pmc_index_term_map (pmid, termid) values \
                        (?, ?)', (dataSet.IDList[index], termid[0]))
                    self.dbconnect.commit()
                else:
                    self.sqlstmt = "select termid from pmc_index_list\
                        where term = '%s'" % term
                    self.dbcursor.execute(self.sqlstmt)
                    termid = self.dbcursor.fetchone()
                    self.sqlstmt = 'select pmid from \
                        pmc_index_term_map where termid = %s and \
                        pmid = %s' % (termid[0], \
                        dataSet.IDList[index])
                    self.dbcursor.execute(self.sqlstmt)
                    if len(self.dbcursor.fetchall( )) == 0:
                        self.dbcursor.execute('insert into \
                            pmc_index_term_map (pmid, termid) \
                            values (?, ?)', (dataSet.IDList[index],
                                termid[0]))
                        self.dbconnect.commit()
            except: self.dbconnect.rollback()
        else:
            try:
                self.dbcursor.execute('insert into pmc_abstract (pmid, \
                    text, ablen) values (?, ?, ?)', \
                    (dataSet.IDList[index], \
                    str(dataSet.abstractList[index]), \
                    len(dataSet.abstractList[index])))
                if len(dataSet.titleList[index]) > 150:
                    dataSet.titleList[index] =
                        dataSet.titleList[index][0:150]
                if len(dataSet.volumeList[index]) > 5:
                    dataSet.volumeList[index] =
                        dataSet.volumeList[index][0:5]
                self.dbcursor.execute('insert into pmc_article_info \
                    (pmid, title, issn, issue, volume, pagination, doi)\
                    values (?, ?, ?, ?, ?, ?, ?)', \
                    (dataSet.IDList[index], dataSet.titleList[index], \
                    dataSet.ISSNList[index], '0', \
                    str(dataSet.volumeList[index]), \
                    str(dataSet.pageList[index]), \
                    str(dataSet.DOIList[index])))
                for authorL in dataSet.authorList[index]:
                    for author in authorL:
                        try:
                            self.dbcursor.execute('insert into pmc_author\
                                (pmid, lastname, initial, forename) values \
                                (?, ?, ?, ?)', (dataSet.IDList[index], \
                                author['LastName'], author['Initials'], \
                                author['ForeName']))
                        except TypeError:
                            if type(author) == 'string' and
                                len(author) > 0:
                                    self.dbcursor.execute('insert into \
                                        pmc_author (pmid, lastname, initial,\
                                        forename) values (?, ?, ?, ?)', \
                                        (dataSet.IDList[index], author, '', ''))
                self.dbconnect.commit()
            for term in searchTerm:
                if (self._checkTerm(term, 'pmc_index_list') ==

```



```

'FALSE'):
    self.dbcursor.execute('insert into pmc_index_list\
        (term, indextype) values (?, ?)', (str(term),
        'all_fields'))
    self.sqlstmt = "select termid from pmc_index_list\
        where term = '%s'" % term
    self.dbcursor.execute(self.sqlstmt)
    termid = self.dbcursor.fetchone()
    self.dbcursor.execute('insert into \
        pmc_index_term_map (pmid, termid) values \
        (?, ?)', (dataSet.IDList[index], termid[0]))
    self.dbconnect.commit()
else:
    self.sqlstmt = "select termid from pmc_index_list\
        where term = '%s'" % term
    self.dbcursor.execute(self.sqlstmt)
    termid = self.dbcursor.fetchone()
    self.sqlstmt = 'select pmid from \
        pmc_index_term_map where termid = %s and \
        pmid= %s' % (termid[0], dataSet.IDList[index])
    self.dbcursor.execute(self.sqlstmt)
    if len(self.dbcursor.fetchall( )) == 0: #no link
        self.dbcursor.execute('insert into \
            pmc_index_term_map (pmid, termid) values \
            (?, ?)', (dataSet.IDList[index], termid[0]))
        self.dbconnect.commit()
    except: self.dbconnect.rollback()
    if (process_count % 5000) == 0: self.dbconnect.commit()
self.dbconnect.commit()

```

## A.4. Other Selected Source Codes

### Code for Co-Expression Network, NLP-derived Network and Co-Occurrence Network Analysis (Chapter 4)

```

import sets

lc = open('lc_cor.csv', 'r').readlines()
lc = [x[:-1] for x in lc]
lc = [x.split('\t') for x in lc]
lc = [[x[0], x[1], float(x[2])] for x in lc]
l = [(x[0], x[1]) for x in lc]
l = sets.Set(l)

def process_sif(file):
    a = open(file, 'r').readlines()
    a = [x[:-1] for x in a]
    a = [x.split('\tpp\t') for x in a]
    return [(x[0], x[1]) for x in a]

a = sets.Set(process_sif('activate.sif'))

print "# intersect of activate.sif and LC data: " + \
    str(len(l.intersection(a)))
print "# LC data not in activate.sif: " + str(len(l.difference(a)))
print "# activate.sif not in LC data: " + str(len(a.difference(l)))
print ""

a = sets.Set(process_sif('bind.sif'))

print "# intersect of bind.sif and LC data: " + str(len(l.intersection(a)))
print "# LC data not in bind.sif: " + str(len(l.difference(a)))
print "# bind.sif not in LC data: " + str(len(a.difference(l)))
print ""

a = sets.Set(process_sif('pubgenel.sif'))

print "# intersect of pubgenel.sif and LC data: " + \
    str(len(l.intersection(a)))
print "# LC data not in pubgenel.sif: " + str(len(l.difference(a)))
print "# pubgenel.sif not in LC data: " + str(len(a.difference(l)))
print ""

```

```

cor = 0.74
while (cor < 1.0):
    t = [(x[0], x[1]) for x in lc if x[2] > cor]
    l = sets.Set(t)
    cor = cor + 0.01
    print "LC correlation: " + str(cor)
    print "# intersect of pubgene1.sif and LC data: " +
str(len(l.intersection(a)))
    print "# LC data not in pubgene1.sif: " + str(len(l.difference(a)))
    print "# pubgene1.sif not in LC data: " + str(len(a.difference(l)))
    print ""

l = sets.Set(process_sif('all_ma_intersect.sif'))
a = sets.Set(process_sif('activate.sif'))

print "# intersect of activate.sif and intersected data: " +
str(len(l.intersection(a)))
print "# intersected data not in activate.sif: " + str(len(l.difference(a)))
print "# activate.sif not in intersected data: " + str(len(a.difference(l)))
print ""

a = sets.Set(process_sif('bind.sif'))

print "# intersect of bind.sif and intersected data: " +
str(len(l.intersection(a)))
print "# intersected data not in bind.sif: " + str(len(l.difference(a)))
print "# bind.sif not in intersected data: " + str(len(a.difference(l)))
print ""

a = sets.Set(process_sif('pubgene1.sif'))

print "# intersect of pubgene1.sif and intersected data: " +
str(len(l.intersection(a)))
print "# intersected data not in pubgene1.sif: " + str(len(l.difference(a)))
print "# pubgene1.sif not in intersected data: " + str(len(a.difference(l)))
print ""

```