# Automatic Summarization of Mouse Gene Information for Microarray Analysis by Functional Gene Clustering and Ranking of Sentences in MEDLINE Abstracts

By

Jianji Yang, MS

A Dissertation

Presented to the Department of
Medical Informatics and Clinical Epidemiology
and Oregon Health & Science University
School of Medicine
in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy
In
Biomedical Informatics

June, 2007

The dissertation “Automatic Summarization of Mouse Gene Information for Microarray Analysis by Functional Gene Clustering and Ranking of Sentences in MEDLINE Abstracts” by Jianji Yang has been examined and approved by the following Examination Committee:

Dr. William Hersh, Dissertation Advisor
Professor, Oregon Health & Science University

Dr. John Belknap, Committee Member
Professor, Oregon Health & Science University

Dr. Aaron Cohen, Committee Member
Assistant Professor, Oregon Health and Science University

Dr. Shannon McWeeney, Committee Member
Assistant Professor, Oregon Health & Science University

Dr. Brian Roark, Committee Member
Assistant Professor, OGI School of Science & Engineering, Oregon Health & Science University

# TABLE OF CONTENTS

# List of Tables

# List of Figures

# Acknowledgements

My sincere thanks to my committee members that made this dissertation possible: to Dr. Bill Hersh, my advisor, for his guidance, inspiration, and unwavering support; to Dr. Aaron Cohen, for his encouragement and generous help; to Drs. John Belknap, Shannon McWeeney and Brian Roark for sharing their expertise in biology, biostatistics and language processing. I always felt that I was so lucky to have such a great team for my dissertation committee.

Next, I owe thanks to National Library of Medicine's fellowship program (NLM Training Grant 1T15 LM009461.), which gave me the financial support for these productive four years.

I would also like to thank the OHSU genetics researchers who spent their precious time to help me evaluate the system. Their insight and suggestions were invaluable to the project.

Furthermore, I would like to express my gratitude to my friends in the OHSU Biomedical Informatics program, whose friendships were invaluable and kept me going through the whole project.

Finally, special thanks to my family, Yizhi, Bill, and Kevin Wang, for their support during these four years of graduate study.

# Abstract

Tools to automatically summarize gene information from the literature have the potential to help genomics researchers better interpret gene expression data and investigate biological pathways. Even though several useful human-curated databases of information about genes already exist, these have significant limitations. First, their construction requires intensive human labor. Second, curation of genes lags behind the rapid publication rate of new research and discoveries. Finally, most of the curated knowledge is limited to information on single genes. As such, most original and up-to-date knowledge on genes can only be found in the immense amount of unstructured, free text biomedical literature. Genomic researchers frequently encounter the task of finding information on sets of differentially expressed genes from the results of common high-throughput technologies like microarray experiments. However, finding information on a set of genes by manually searching and scanning the literature is a time-consuming and daunting task for scientists. For example, PubMed, the first choice of literature research for biologists, usually returns hundreds of references for a search on a single gene in reverse chronological order. Therefore, a tool to summarize the available textual information on genes could be a valuable tool for scientists. In this study, we adapted automatic summarization technologies to the biomedical domain to build a query-based, task-specific automatic summarizer of information on mouse genes studied in microarray experiments - mouse Gene Information Clustering and Summarization System (GICSS). GICSS first clusters a set of differentially expressed genes by Medical Subject Heading (MeSH), Gene Ontology (GO), and free text features into functionally similar groups;

next it presents summaries for each gene as ranked sentences extracted from MEDLINE abstracts, with the ranking emphasizing the relation between genes, similarity to the function cluster it belongs to, and recency. GICSS is available as a web application with links to the PubMed (www.pubmed.gov) website for each extracted sentence. It integrates two related steps, functional gene clustering and gene information gathering, of the microarray data analysis process. The information from the clustering step was used to construct the context for summarization. The evaluation of the system was conducted with scientists who were analyzing their real microarray datasets. The evaluation results showed that GICSS can provide meaningful clusters for real users in the genomic research area. In addition, the results also indicated that presenting sentences in the abstract can provide more important information to the user than just showing the title in the default PubMed format. Both domain-specific and non-domain-specific terminologies contributed in the informative sentences selection. Summarization may serve as a useful tool to help scientists to access information at the time of microarray data analysis. Further research includes setting up the automatic update of MEDLINE records; extending and fine-tuning of the feature parameters for sentence scoring using the available evaluation data; and expanding GICSS to incorporate textual information from other species. Finally, dissemination and integration of GICSS into the current workflow of the microarray analysis process will help to make GICSS a truly useful tool for the targeted users, biomedical genomics researchers.

# Chapter 1. Introduction

With the increasing volume of published on-line full-text scientific articles, even the most robust Information Retrieval (IR) system returns more documents and abstracts than biomedical scientists are able to manually review. The problem is aggravated by the information-intensive nature of "high-throughput" technologies (e.g., microarray experiments) that can study expression in a given biologic context at a genome-wide scale. In fact, these advanced technologies and the increasing number of publications discussing genomic findings impair our ability to fully comprehend the meaning of the information that is embedded in the vast body of free text biomedical literature. As such, the ability to use the literature to interpret the results of the experiments at hand is limited [1]. Hence, tools that are able to survey the large quantity of literature can be helpful to the scientists interpreting and planning these large scale genome-wide microarray experiments.

## *1.1 Introduction to microarray technology*

Microarrays or gene chips are microscopic arrays of DNA spots (each usually representing one gene) spatially arranged and attached to a solid surface, such as glass, plastic or silicon chip. When the chip is hybridized with fluorescence-labeled cDNA made from mRNA extracted from cells, the abundance of different mRNA molecules in the cells can be measured by detecting the fluorescence levels of each spot on the chip. Microarray technology [2] is often used for gene expression profiling in many areas of biomedical research to measure the abundance of messenger ribonucleic acid (mRNA) transcripts in genome wide scale. While the first microarrays available contained 6000 genes of the yeast genome in 1997, Affymetrix[1] now has commercial microarray chips that represent the entire human genome, more than 30,000 genes. A microarray is typically a glass slide with tens of thousands of spots that each contains identical DNA oligonucleotides that are fragments from a known gene sequence. Currently the sequences on the arrays come from genome sequencing project. The technology can be used to investigate the differential expression levels of genes in the whole genome under different conditions, e.g. control vs. diseased, young vs. aged, or different cell types *etc*. For instance, experiments can be performed to conduct comparison of gene expression between normal and breast cancer tissues. The mRNAs from the normal and cancerous tissues differentially bind to the complementary nucleic acid sequences on the array and abundance of mRNA in both tissues is revealed. Since these experiments can measure the expression level of tens and thousands of genes simultaneously, the analysis of the results produced is nontrivial because of the large data size. Even the differentially expressed gene list is usually comprised of hundreds of genes.

## *1.2 Motivating example*

The following is a motivating example that illustrates the information overload problem scientists face when they are searching information on genes during the analysis phrase of a microarray experiment. It also demonstrates a possible use of the GICSS system

[1] http://www.affymetrix.com/index.affx

implemented in this project. “Dr. Smith” is a senior behavioral neuroscientist at OHSU. He studies genes that are associated with the alcohol addiction trait in mice. In one microarray experiment, he compared the level of thousands of mRNAs in normal and alcoholic mice. The experiment returned hundreds of genes that show differentiated levels of expression between these two types of mice. He then assembled an Excel spreadsheet with annotations (in Gene Ontology terminology) for each gene from the chip maker, Affymetrix. Currently, he would use online resources, such as PubMed, the Mouse Genome Informatics (MGI) database, and his knowledge in this area to construct the relationships between the expression patterns and the functional groups of genes in his results. The problems with this approach are:

1. The access to the literature is generic. The conventional method for doing this is to search the literature one gene or one gene group at a time. This method may work well at a small scale, but for hundreds of genes, it is a labor-intensive task to simultaneously analyze their roles in the cell process and the interrelations among them.
2. The search engine returns thousands of citations in reverse chronological order.
3. The structured databases have curated information for each single gene. But there is no explicit information linked to this particular *set* of genes obtained from this experiment. For example, there is no information on how these set of genes are related functionally and how likely they involve in similar cellular process.
4. Scientists are on their own to assemble the big picture from a large collection of information sources including the pattern of gene expression profile, functional

annotation for each gene, and vast amount of literatures discussing the subsets of differentially expressed genes.

GICSS, the two-step summarization system built in this project, aims to help at this stage in Dr. Smith’s analysis. With GICSS, Dr. Smith is able to enter his list of up-regulated and down-regulated gene lists, and get back clusters of genes according to their functional similarity. He can see that, for example, a cluster of genes involved in dopamine metabolism are up-regulated (as highlighted in red in the display) in the alcoholic mice versus wild types. He believes this is interesting and decides to pursue further by exploring the literature. GICSS simplifies the task of searching and scanning the literature by providing summaries on the genes by sentence extraction. He can click on one of the genes in a cluster of interest and the GICSS system shows him ranked sentences that relate to that gene and its function within the context of the cluster. He then can scan the sentences and follow links to PubMed article abstracts describing how the gene is related to the dopamine receptor. He now has functionally structured gene clusters instead of a flat list, and access to literature filtered for sentences related to functional relationships among genes. Therefore, he can spend less time scanning returned abstracts (which on average have 10-15 sentences) from PubMed. In addition, the time spent reviewing abstracts is more fruitful because it is highly tailored to his specific information needs. The time saved can be of more constructive use, such as designing further experiments.

Sections 1.3 and 1.4 below provide an overview of prior research work on tools that process the literature for information (even though most of them have not been used routinely by biologists), in addition to available knowledge resources that are available for scientists to  use in analyzing their results.

## *1.3* ***Curated databases***

Much effort has been put into creating resources for structuring information on genes and pathways. Most of these resources are created by manually extracting information from publications of scientific research. They require intensive human labor and for that reason, usually it takes a while for the up-to-date information from the literature to be curated and entered into the databases. Some of the notable resources that scientists working with microarray data frequently used are informatics databases for specific species, e.g., Mouse Genome Informatics[2] for mice, and FlyBase (A Database of Drosophila Genes & Genomes)[3] for Drosophila. They provide information on genes for that particular species, such as sequence, functional annotation, genome maps, and phenotypic information. Another curated information source is the bioinformatics suite from the National Center for Biotechnology Information (NCBI)[4], including GenBank, Entrez Gene (a searchable database of gene information) and Gene Expression Omnibus (a gene expression/molecular abundance repository with expression data browsing, query

[2] http://www.informatics.jax.org/

[3] http://flybase.bio.indiana.edu/

[4] http://www.ncbi.nlm.nih.gov/

and retrieval). The Source database from Stanford[5] collects and compiles data from many publicly available data sources to provide consolidated information for genes in GeneReports.

Moving up from the individual gene functional annotation resources to gene group, pathway and process level, there are also repositories of information on how each individual gene participates in molecular interaction pathways and cell processes. One of these resources is Pathway Database in Kyoto Encyclopedia of Genes and Genomes (KEGG)[6]. It has manually drawn maps and annotations of interaction and reaction networks for the well-studied areas of system biology, such as, metabolism, genetic information processing, and disease processes.

These resources are very valuable in providing information on a particular gene and well-known pathways. On the other hand, the building and maintenance of these databases require much human labor. As such, the information provided sometimes lags behind the rapidly evolving scientific knowledge, and biologists still have the need to go to free-text publication for original, up-to-date information and evidence.

## *1.4 Information from the literature*

In addition to curated databases, biologists search related literature for original and up-to-date information. The first choice of search is National Library of Medicine (NLM)'s

[5] http://source.stanford.edu

[6] http://www.genome.jp/kegg

PubMed literature database - NLM's usage statistics data showed that the rate of increase in searches to PubMed has been between 11% - 35% over the last five years and in the year 2006 PubMed recieved an average of over two million searches per day. On the other hand, with the advent of both the biological technologies and the efficiency of on-line publishing, the number of potentially relevant articles continues to increase rapidly. MEDLINE added over 623,000 citations in 2006 and had 15,940,559 total records as of July 2, 2007 (data from NLM's MEDLINE/PubMed resources guide[7]). As a result, even the most robust IR engine returns more documents and abstracts than biomedical scientists are able to manually review. A simple PubMed search on a gene symbol will return thousands of hits. For example, a query 'NR1' (N-methyl-D-aspartate (NMDA) receptor subunit 1) submitted on July 9, 2007 to PubMed retrieved a total of 1,719 matches.

To facilitate this information searching process, many efforts have been put into building text-processing tools to uncover the knowledge buried in the literature, with varying success. In general, adapting text-processing technologies in the biomedical domain has been slowed by major challenges [1], such as non-standard nomenclatures for genes, proteins and other biological entities; domain specific languages; the highly complex interrelation within biological systems; and the lack of standard ontology in the domain. Despite these difficulties, some efforts have achieved some amount of success and several of them start to receive real-world use. These approaches usually involve using

[7] http://www.nlm.nih.gov/bsd/revup/revup_pub.html#med_update

combinations of NLP and text-mining techniques and examples of these approaches include: document clustering, text classification, information extraction, question answering and summarization.

Document clustering techniques attempt to group a text collection into clusters of articles that relate to a similar topic. A example of using document clustering in biomedical research is a system called PubClust [3] that groups the result of any PubMed search using words in the returned abstracts as features so that users can pick the topics of interest for their purpose.

While document clustering uses unsupervised learning techniques, text classification employs supervised learning techniques to label natural language texts with thematic tags from a set of predefined categories. In TREC 2005, one of the sub-tracks in genomic track was to triage biological texts into four categories [4].

Information extraction (IE) methods discover structured information from free text using NLP techniques, lexical resources and semantic constraints. In the biomedical domain, IE is used mostly to extract relations and specific facts about biological entities[5,6]. IE often involves hand-crafted templates and rules based on expert knowledge and intensive NLP processing with high computational complexity.

Question answering is another technology to help user to get to the relevant information quickly and has been getting more attention recently. The idea is to let users ask a structured question, such as '*What is the role of prion in mad cow disease?*' and have the system process the document collection to extract the corresponding information from a text source to provide an answer. This is similar to IE but is real-time and gives the user more control over the information extracted as well as more context with which to verify and apply the generated answers. The TREC Genomics Track has recently focused on this task [7].

Another potentially useful, but less-studied approach is to automatically produce customized summaries for information related to a specific user information need. For this project in particular, the potential users are scientists who are analyzing the result of a given microarray experiment, so information on genes that are differentially expressed under the different experimental conditions and their relations are of importance. Summarization is defined by Sparck Jones [8] as "*a reductive transformation of source text to summary text through content reduction selection and/or generalization on what is important in the source*". Automatic summarization systems have been studied since the late 1950s [9,10] and applied in different domains such as news, with some notable success [11]. However, adopting the technology in the biomedical domain is not straightforward. There are fewer resources available in biomedicine, such as test corpora and knowledge bases, which makes training and evaluation more difficult. Summaries for biomedical literature probably require a different focus. The information that most interests scientists may reside in sentences describing some specific biological processes (use of domain specific language e.g. phosphorylation, activation, co-expression) while in the news domain, the *who, when, what, and where* elements are generally applicable and often the most important [12]. These specific information requirements can be exploited in the biomedical domain by emphasizing domain-specific keywords to extract important information and to construct summaries.

## *1.5 General automatic summarization*

This section provides a basic introduction to the field of automatic summarization. First, major concepts and definitions in the field of automatic summarization are introduced. Then, a brief history of automatic summarization with discussion of different approaches to construct summaries is described. The approach used in GICSS was discussed in the context of automatic summarization in general.

### 1.5.1 What is automatic summarization?

As the amount of published and on-line information increase, generation of condensed text that summarizes the vast amount of documents for human consumption is one way to help us digest the information and find the obscured task-relevant information. A summary as defined by Sparck-Jones [8] can be generated by a human or computer system. In the latter case, the computerized system is called an automatic summarizer and the process is called *automatic summarization*.

There are many types of summary as categorized on different axis. Categorized by purpose, *indicative summaries* provide a general idea of the original text subject matter but without specific content; while *informative summaries* cover the salient information in the source to some level. In addition, *critical summaries* evaluate the source text and express the abstractors' view.

Categorized by form of content, summaries generated by *extraction* consist entirely of material (words, sentences and/or paragraphs) copied from the source. Extraction serves to identify the more important and distinct portions of the source material. On the other hand, summaries by *abstraction* contain novel material that is generated from the source texts. Abstraction involves inference from the source text and uses references to background information. Abstraction can produce summaries with a higher degree of compression, i.e. shorter summaries. On the other hand, because abstraction requires deeper analysis, usually at the semantic level, and a wealth of knowledge to draw upon, computational tractability is an issue for real-time systems. Furthermore, since abstraction generates novel materials, it also carries the risk of mis-inference and falsely 'inventing' information that is not in the source text.

Categorized by dimensions, summaries can be generated from a *single document* or *multiple documents*. Usually multi-document summarization has additional challenges in addition to all of those presented in single document summarization. These include co-reference resolution across documents, higher compression rate, redundancy reduction and confliction identification.

Categorized by context, a summary can be based on source texts retrieved by a query the user entered or query-independent. By genre, a system can be a *generic summarizer*, which can generate summaries for source texts of any field; or it can be a *specific domain summarizer*, which takes advantage of the available knowledge and special format and structure of texts of a specific domain to achieve better summaries.

### 1.5.2 Brief history of automatic summarization and major approaches

Summarization processes typically involve the following three steps [13]:

1. Analysis: analyze and build a content representation of the source texts

2. Transformation: map the content representation into summary representation
3. Synthesis: generate and output summary from the above representation

The following section presents some of the key work in the field and the current state of the art. Even though their approaches may be different in many areas, most of them follow the above three-step paradigm.

Most of the early systems of automatic summarization from the late 1950s to the 1970s were single document summarizers by sentence extraction. They used shallow features including word frequency, position, cue phrases, and theme terms to determine the importance of the extraction unit (mostly sentences, and some used paragraphs). A score was calculated for each feature and the scores were normalized and summed. The text units with the highest summed scores were presented as summary. This type of summarizers included Luhn [9] and Edmundson [10]'s work in sentence extraction summarization. In Edmundson's paradigm, the weight of each sentence was calculated as a linear combination of four features:

*Weight(s) = a* CuePhrase(s) + b * AddTerm(s) + c * ThematicTerm(s) + d * Location(s)* where

- *CuePhrase(s):* Lexical or phrasal with summary cues: positive weights for bonus words ("significant", "confirms", etc.), negative weights for stigma words ("hardly", "impossible", etc.)
- *AddTerm:* Weight assigned to a sentence for terms in it that are also present in the title, headline, initial paragraph, or the user's profile or query

- *ThematicTerm:* The presence of statistically salient terms (e.g., *tf.idf* terms) in a sentence, based on Thematic Term Assumption [9] that high frequency content words are positively relevant.
- *Location:* Sentence's location within the document - beginning, middle or end of a paragraph or the entire document, or whether it occurs in prominent sections such as the document's introduction or conclusion

Many measurements these early work introduced, such as key terms and cue words, are still used to date.

Kupiec et at. [14] extended the paradigm by Edmundson and introduced machine learning approaches for extraction with a naïve Bayes classifier. Further utilization of machine learning techniques included clustering text to achieve diversity by Nomoto [15] and using Hidden Markov Models for text selection by Conroy [16].

In addition to the above statistical approaches to identify the salient part of the source texts, various methods focused on constructing summaries by exploiting the discourse structure of the text. Marcu [17,18] studied the nucleus and satellite relations of the text structure based on the rhetorical structure theory by Mann [19]. Another notable approach is lexical chain, a sequence of related words in the text that represents a cohesive structure and specific topic of the text. Barzilay [20] introduced a way to calculate the score of lexical chains using the length and homogeneity index based on WordNet[8] as a means to identify salient part of the source text.

[8] http://wordnet.princeton.edu/

The above methods are extraction type summarization. Parallel efforts have been working on summarization through abstraction. Early approaches mainly focused on template-filling type abstraction, using templates as the knowledge and semantic base required for abstraction. The creation of these templates usually required extensive human labor and the templates were often restricted to very specific topics and domains. Examples of this type of systems include the FRUMP news summarizer by DeJong [21], and SCISOR [22] summarization of corporate mergers and acquisitions news. Current advances in digitalized domain-specific knowledge base enabled more sophisticated abstraction involving generalization, inference and exploiting semantic relations between concepts. Fiszman et al. took advantage of the UMLS[9] Specialist Lexicon and Semantic Network to construct summaries of the A.D.A.M. © [10] online encyclopedia [23].

Summarization on multiple documents has additional technical challenges such as requiring higher compression rates, redundancy elimination, contradiction identification and co-reference resolution across documents. Maximal marginal relevance (MMR) was used by Carbonell and Goldstein [24] to emphasize novelty of selected information, a way to handle redundancy. Radev et al. used a centroid-based clustering technique to improve diversity of the sentences selected for summary [11].

[9] http://www.nlm.nih.gov/research/umls/about_umls.html

[10] http://www.adam.com/

### 1.5.3 What type of summarization is used in this project?

The GICSS system built in this project is in the category of multi-document summarization by sentence extraction. It is query-based and takes in a set of genes from the result of a microarray experiment. It is a summarizer that operates on a given question within a specific domain, i.e. genomic research in biomedicine. We used a statistical approach based strongly on Edmundson's paradigm with modifications exploiting the domain-specific terminologies and entities appropriate to the problem space.

# Chapter 2. Related Work

This chapter describes the previous work specifically related to this project. They are grouped into two categories: functional gene clustering by features other than expression profile, and summarization on gene-related information.

## *2.1 Gene clustering*

Results of a microarray experiment have the expression levels of each gene/transcript under the different biological conditions as studied in the experiment. Assuming that genes with similar functions or within the same biological pathways will have similar expression patterns, cluster analysis of gene expression profile is one of the essential components of exploratory analysis of all microarray datasets. While gene clustering by expression profile is of great value to investigators, in this project the focus is on gene clustering from a different perspective, i.e., how the genes that are found to be differentially expressed in the experiment cluster according to previous-known knowledge as represented in the literature or in the curated database. In other words, we are to find gene clusters using features such as Medical Subject Heading (MeSH)[11] headings, Gene Ontology (GO)[12] annotations, and free text in scientific literature.

A number of approaches to find functional gene groups by analysis of literature profile have been proposed. Masys et al. [25] identified gene groups based on co-occurrence of

[11] http://www.nlm.nih.gov/mesh/

[12] http://www.geneontology.org/

MeSH terms in MEDLINE citations. PubGene described by Jenssen et al. [26] used gene name co-occurrence in MEDLINE abstracts to identify related gene neighbors and build gene relation networks. Chaussabel and Sher [27] clustered genes by analyzing the occurrence of a filtered list of terms in MELINE abstracts, generating literature gene 'heat map' similar to the one generated by analysis of expression profiles. Glenisson [28,29] explored the use of 'bag of word' vector space representation of literature profile for functional gene clustering. The text sources used in their TXTgate application [29] are from selected annotation fields and linked MEDLINE abstracts in the curated repositories LocusLink and the *Saccharomyces* Genome Database (SGD).

There are also works focusing on comparing the effectiveness of different algorithms in gene clustering. Homayouni et al [30,31] found that Latent Semantic Indexing(LSI) can be a robust method to discover gene relationships. Liu [31] tested an approach called the Bond Energy Algorithm (BEA), originally used in clustering questions in psychological research instruments. The results suggested BEA compared favorably to other popular clustering algorithms, such as hierarchical, k-means and Self Organizing Maps (SOM).

There are also works on combining the information in literature with the gene expression profile to generate gene clusters. Raychaudhuri et al [32,33] used information in the literature to fine-tune the boundary of clusters found in expression profile analyses. An algorithm developed by Kuffner et al [34] combined both microarray expression data and MeSH terms and words in MEDLINE abstracts to identify gene clusters with corresponding literature topics. Huang et al [35] modified the expression-based gene distance metric by shrinking the distance to zero if genes share functional keywords.

This literature-informed metric was used to generate the final clusters with improved results over standard method.

All of the above approaches focused only on the clustering process and clusters generated from the process were their final results, while GICSS system attempts to utilize the result of the clustering process in the next step—summarization of gene information. The evaluation of the clustering algorithms usually involved using distinct gene groups from certain cell cycles or GO term branches as test gene sets. The effectiveness of the algorithms was judged by the ability to correctly put the genes in the right groups and use hard statistics like internal similarity, mutual information and entropy as measurement [31,36,37]. This approach was straightforward and gave quantitative results, but there were some shortcomings too. Distinct gene groups are much easier to cluster than real gene sets from microarray experiment because distinct gene groups usually have longer distance between the groups. In addition, how do the quantitative values correlate with the meaningfulness of the cluster in the analysis process is still an under-studied question.

## 2.2 ***Summarization in biomedical domain***

Discovering functional related gene clusters is only one of the initial steps of the microarray analysis process. After identifying some interesting clusters, scientists will then focus on the genes in the clusters and try to elucidate how these related gene groups contribute to the conditions and contexts studied in the experiment. It is very common that scientists will encounter many unfamiliar genes when they study the clusters. (This assumption was confirmed during the system evaluation study as discussed later.) Hence, the scientists will engage in searching for gene information in the context of the shared functions discovered in the clustering process. Therefore, the functional gene clustering and information searching are closely related parts of microarray data analysis. As mentioned in the last chapter, automatic summarization can be used as a potential tool to facilitate the access information in the rapidly increasing amount of free text literature. Work in biomedical information summarization has been mostly of the extraction type. One approach is keyword summary. Domain standard terminologies, such as MeSH headings, Gene Ontology terms, are usually the choice of terms for keywords. MedMeSH [38] can take in a gene cluster discovered in cluster analysis of the expression profile and retrieve citations for each gene from the MEDLINE database. MeSH terms for the article are extracted from the citation and then the statistical distribution patterns are analyzed and compared. Important MeSH terms are then assigned to describe the gene set, i.e., using MeSH terms as keywords to capture the biological significance of the set of genes. This approach can only provide summaries for a cluster of genes, with no further information presented for each gene.

The other approach is sentence extraction. Most of them have expert-defined categories. The computation usually involves machine learning algorithms to classify sentences into these predefined buckets or rely heavily on advanced NLP techniques to match sentence templates. MedMiner [39] is one of the engines that can search the literature and extract sentences about gene-gene interactions. It is limited to discovering relations between two genes only and requires many iterations if used with microarray data which mostly have a list of over a hundred genes. It returns all of the extracted sentences from the articles that demonstrated the relation, without providing more summarization, such as re-ranking to present more relevant sentences first. In this sense, this system can be viewed more as an information retrieval and extraction engine. The GICSS system is customized for the input of the large gene set (usually in the hundreds) from microarray experiment and processes them as a whole instead of pairs of genes.

Another sentence extraction application, BioIE [40], is a rule-based system to extract sentences pertinent to protein information in five predefined classes: *structure, function, disease & therapeutic compounds, localization, and familial relationships.* METIS [41] uses information in Swiss-Prot[13] to generate protein reports and extracts informative sentences from literature using both machine learning and rule-based algorithms. METIS uses the same five predefined categories as BioIE in the learning and sentence extraction process. These two systems use hand-crafted rules, have predefined classes, and the latter also requires training data. These two systems are geared for protein information and the predefined categories reflected this priority so that they are not very useful for getting information for gene lists. In addition, hand-crafted rules required much human effort and they are also hard to update. Furthermore, the predefined categories also limit the information the systems can present and sentences do not fall into these categories will be ignored.

An application by Ling [7] extracts top-ranked sentences about genes from MEDLINE abstracts in the six predefined areas of interest using cosine similarity scoring against the

[13] Swiss-Prot website: http://www.ebi.ac.uk/swissprot/

*six classes: gene products; expression location; sequence information; wild-type function and phenotypic information; mutant phenotype; and genetical interaction.* Even though this approach can provide the basic information for the gene in the above categories, other useful information outside these six categories will be missed out, just like BioIE and METIS. The information is generic about the genes and not specific to the microarray results, i.e. each gene summary is independent to others. GICSS system takes in input from the clustering algorithm, therefore includes some context information from the closely related genes. Further more, all three systems required some level of training and while training data was hard to get, retraining is another issue that can affect future performance.

The list of selected related work in functional gene clustering and information summarization described in this section is presented in Table 1, with feature comparison to each other and to the GICSS system.

**Table 1. List of related works in gene clustering and summarization as compared to the GICSS system. It shows that most previous work has focused on a single one of these two areas and that most gene or protein information summarization systems had predefined information categories.**

| **Systems** | **Algorithm highlight** | **Input** | **Features used** | **Categories of information** | **Level of summarization** |
|---|---|---|---|---|---|
| Masys *et al* [25] | gene clustering | Any gene set | MeSH co-occurrence | No | Unspecified |
| PubGene [26] | gene clustering | Any gene set | Gene name co-occurrence | No | Unspecified |
| Chaussabel and Sher [27] | gene clustering | Any gene set | Filtered terms from MEDLINE abstracts | No | Unspecified |
| Kuffner [34] | gene clustering | Full gene chip | MeSH, text words, expression data | No | keyword |
| PubClust [3] | search result clustering | PubMed result | text words | No | keyword |
| MedMeSH [38] | gene cluster summary | Gene cluster | MeSH terms | No | keyword |
| Ling [42] | gene summary by extraction | one gene name | text words, training data | Yes | sentence |
| BioIE [40] | protein summary by extraction | protein name | text words, hand-crafted templates | Yes | sentence |
| METIS [41] | protein summary by extraction | protein name | text words, templates and training data | Yes | sentence |
| MedMiner [39] | gene relation information extraction | two genes | text words | No | sentence |
| GICSS system | gene clustering and summary | gene set with fold change | MeSH, GO terms, text words | yes | keyword and sentence |

## *2.3 Related Work Summary*

As described in the previous section, the work in gene clustering and summarization are two different approaches even though the work flow for the experiment analysis is integrated and closely related. Work in the area of gene clustering focus on perfecting the algorithm, feature selection and the summarization effort limited to keyword summary for the generated clusters. On the other hand, work in the area of gene information summarization focus on extraction of certain types of information for a single gene. The information gathered from the clustering result (keywords and closed related genes) is has not been previously used in the summarization process. The summaries are usually generic and independently generated without considering the relation to other genes, which is of interest in microarray analysis. In addition, most of the related work either depend heavily on hand-crafted templates and NLP patterns or require extensive training data. Furthermore, most work on summarization have several predefined information categories, but may miss information outside these categories. Finally, there was no *experimental evaluation* of the systems with scientists using real data from microarray experiments they are analyzing. For example, clustering genes taken from several independent known pathways will be very different from clustering genes from the raw list of microarray output.

Thereform this study extends and is distinct from prior work in several ways:

1. The GICSS system focuses on helping scientists search for supporting evidence when they are analyzing a microarray expression profile. It integrates the two closely related steps in the analysis process, clustering and literature summarization. Genes

are first clustered into functionally related groups, which are important especially when the size of the gene set is large. The information gathered from this step is used to inform the second step, gene information summarization.

2. There are no predefined categories. Using MeSH terms associated with the publications on the genes, GO terms assigned to the genes and free text from abstracts, the clustering step tries to capture the naturally occurring clusters presented in the literature.
3. There are no labor-intensive handcrafted rules or training data.
4. The summary for each gene is presented in a ranked sentence format. The ranking algorithm emphasizes gene relations, length, similarity to its cluster theme, recency, existence of domain-specific terms and non-domain specific language features. Sentence was chosen as the information unit based on the assumption that the sentence is more informatively intact and richer than keywords, and short enough for a research scientist to quickly go through and decide if reading the full abstract or article is warranted. Furthermore, a study on the effect of information extraction units (abstract, sentence, and phrase) [43] indicated that using sentence granularity achieved the highest effectiveness.
5. The GICSS system was evaluated in experimental setting with scientists working on their own, current microarray experimental data.

6. The generalizability of the approach was demonstrated by expanding it to human genes by substituting a human gene and protein name entity recognition system (NER) in place of the mouse NER system used for the main evaluation.

## *2.4 Research Statements*

*Main research questions*

The goal of this study is to design, build and evaluate an automatic summarization system for information on genes differentially expressed during microarray experiments. The intended end-users include genomic researchers, along with their students and research assistants, who have sets of differentially expressed genes as a result of microarray experiments and need to search for gene information from the literature in the process of analysis.

The two questions this study tries to answer are:

1. After a researcher has obtained the expression pattern of a list of differentially expressed genes, the GICSS system clusters this set of genes into functional groups by drawing from the available literatures. Can the cluster and summary words provide useful information in the context of this particular experiment to help the researcher in analyzing the expression pattern? Sub-questions are:
    1. Are the gene clusters meaningful?
    2. Is there preference for the features (MeSH, GO, and text words) used in the clustering process?

2. In order to facilitate access to the literature, a summary for each gene is constructed by extraction and re-ranking the sentences by importance. By combining the conventional extraction summarization paradigms [44] and special requirements and knowledge of biological domain, can our sentence ranking algorithm facilitate access and exploration of the large amount of original literature by presenting the informative sentences as summaries to the researcher? Sub-questions are:

   1) By presenting sentences in the abstract with reference to genes, can we provide more information than PubMed's standard title presentation, which is the status quo when doing literature search currently?

   2) Furthermore, can our sentence extraction and ranking perform better than just reversed chronological ranking of the same set of sentences with reference to genes?

   3) Finally, how does each feature in the ranking algorithm contribute to the usefulness of the final ranking?

# Chapter 3. Research Design and Methodology

## *3.1 Building the GICSS gene information system*

### 3.1.1 System architecture

The GICSS system was implemented in Python and CGI, and is accessible via the Web. Currently, it is hosted at http://ir.ohsu.edu/jianji/gene_info. The web interface was developed based on the Karrigell web framework[14]. The system architecture is depicted in Figure 1. The system's core components consists of a MeSH term, GO term and word processor, a wrapper around CLUTO[15] (a preexisting application that performs clustering), and a sentence ranker. These three components are explained in detail below.

### 3.1.2 Preprocessing: Extracting gene sentences

The 10-year Medline corpus (from 1994 to 2003) used in TREC 2004 and 2005 Genomics Track was filtered using MeSH Heading "Mice," resulting in a mice subset. In order to achieve higher accuracy in gene name recognition and specific applicability to the mouse researcher user, we decided to focus on mouse genes at this stage. Using a gene and protein name entity recognition and normalization system for mice [45], this subset was processed and gene and protein names were tagged and identified by Mouse Genome Informatics [16] identifiers (MGI-ID). Sentences in abstract and title (treated as a

[14] http://karrigell.sourceforge.net/

[15] http://www-users.cs.umn.edu/~karypis/cluto/index.html

[16] http://www.informatics.jax.org/

sentence) were stored in a database together with other MEDLINE entries, MeSH headings, publication date, journal names. These sentences comprise the text collection used in this study. Sentences containing at least one reference to gene/protein were further indexed by the gene_MGI_ID to facilitate retrieval.

The sentence database has the following three tables:

1. **PMID_DateOfPublication_MeSH (document level information table)**

   PMID -- PMID of MEDLINE records (primary key)

   Dp – date of publication of MEDLINE records

   Mesh – mesh terms of MEDLINE records

2. **MGI_gene_PMID_SentID (gene ID and sentence ID information table)**

   MGI_gene – MGI_ID for mouse genes

   PMID – PMID of the MEDLINE record that has reference to the mouse gene (foreign key)

   SentID – position of the sentence in the abstract that has reference to the mouse gene

   Primary key: MGI_gene, PMID, SentID

3. **PMID_SentID_Sentence ( sentence content table)**

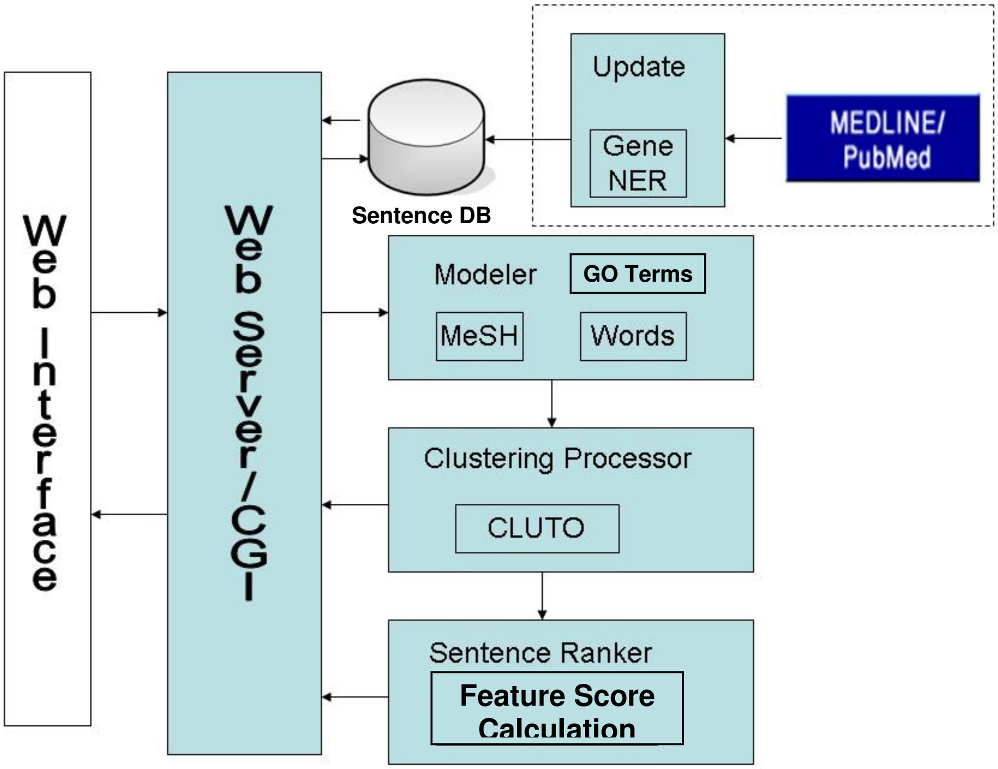
   PMID -- PMID of MEDLINE records (foreign key)

   SentID – position of the sentence in the abstract (foreign key)

   Sentence – sentence text of PMID_SentID

   Primary key: PMID_SentID

**Figure 1**. **System architecture diagram. The GICSS system is composed of a sentence database, three processing modules: gene modeler (gene modeling in language term vector space), clustering processor (a wrapper around CLUTO), and sentence ranker (calculation of sentence score and ranking).**



### 3.1.3 Processing of input gene list

After set of gene names and their expression levels (in the form of fold change) are collected from the user, they are first checked for duplications. The advent of microarray technology allows detection of expression level at transcript level, i.e. spots on the array representing different mRNA sequences transcribed from the same gene (DNA region). There is a chance that a user may input the same gene names more than once with different expression levels, i.e. different transcripts from the same gene having different expression levels in the array. Since currently the system does not have the capacity to process transcript level information, it highlights the duplicates and reminds user of the transcript to gene name mapping issue. More about this topic is discussed in the Limitations and Future Work section. Secondly, the gene names are expanded with synonyms from the dictionary in our gene and protein name entity recognition system [45]. Finally, each gene and its synonyms are mapped to their corresponding MGI-IDs. A summary of this step is then presented to the user before further processing. The information presented includes duplicated genes in the list, gene names that cannot map to any MGI_IDs and for the gene names that can be mapped to MGI_IDs, all the MGI_IDs with links to MGI website.

If the input gene list contains five or less genes, the clustering step described in the next section is skipped. After gene duplication highlight, gene synonym expansion, and MGI_ID mapping, the system goes directly to sentence extraction summarization step.

### 3.1.4 Clustering of genes into functional related groups

The genes are represented by a vector space model with three categories of features.

1. MeSH Headings associated with the publications in which the genes are referenced. The users also have the option to select the MeSH subtrees that are deemed useful to represent gene information to be used. After consultation with a biologist, the default subtrees in the GICSS system used were A-Anatomy, C-Diseases, Chemicals and Drugs, F-Psychiatry and Psychology, G-Biological Sciences, and H-Physical Sciences. Furthermore, terms close to the root are usually very general, for example, the terms: 'GENE' and 'PROTEIN'. Only terms deeper than the second layer are used. For more information on MeSH tree structure, see the MeSH information page at NLM web site[17].
2. GO terms associated with the genes as annotated by MGI obtained from the MGI web site. Terms that indicate any unknown conditions are filtered out, such as '*molecular function unknown'*.
3. Free text words in the sentences with at least one reference to the genes and sentences immediately in before and after them, with stop-words removal (see Appendix) and stemming by Porter stemming algorithm.

Specifically, each gene is modeled as vector of combinations of the above three categories of features. The selection of the categories can be done in the Options web

[17] MeSH http://www.nlm.nih.gov/mesh/

page (Appendix 2). Let M be the number of distinct terms in our collection of N genes, gene *i* is represented as vector:

$G_i = <w_{i1}, w_{i2},\ldots,w_{iM}>$, $w_{i,j}$ is the frequency of *j*th term that is associated with the *i*th gene in the sentence collection for this set of N genes.

The clustering algorithm suite implemented in CLUTO is used for functional gene clustering. The default clustering method is direct k-means. Current specifics of the vector features are listed below:

- If the number of genes is less than six, the clustering step is skipped and the process goes directly to sentence extraction.
- Similarity measure: similarity between genes feature vectors is calculated as the cosine of the angle between the two gene vectors:

  $Cos(gi, gj)=gi \bullet gj/|gi|\,|gj|$
- Number of clusters: this is a parameter CLUTO requires as input. It is determined in run-time by trying different numbers of clusters. Let $\varepsilon$ be the ratio of improvement of internal similarity of all cluster by increasing the number of cluster by 1:

  $\varepsilon = [I2_{(n+1)} - I2_{(n)}]/I2_{(n)}$ where I2 is the measure of internal similarity for all clusters, and n is the number of clusters. When $\varepsilon$ is lower than a certain level, the increase of the number of clusters does not give much increase in the internal similarity for all the clusters, the iteration stops.

ε was set empirically by testing different values. Currently for gene sets with around 100 genes, it is set to 0.035. In other words, n is chosen when the improvement of internal similarity is less than 3.5% by increasing the number of clusters by one. As suggested in post-hoc optimization, the parameter setting may to some extent affect the sentence ranking efficiency due to the contribution of the feature of cluster keywords used in the ranking algorithm. But to study how the parameter setting affects the usefulness of the clusters will require substantial human judgment and was not studied in this project due to limited resource.

- Tfidf is used for scaling the terms. The *term frequency* is $\mathrm{tf_i} = \frac{n_i}{\sum_k n_k}$, where $n_i$ is the number of occurrences of the considered term, and the denominator is the number of occurrences of all terms. The *inverse document frequency* is $\mathrm{idf_i} = \log \frac{|D|}{|\{d : d \ni t_i\}|}$, with |D| : total number of sentences in the corpus and $|\{d : d \ni t_i\}|$: number of sentences where the term $t_i$ appears (that is $n_i \neq 0$). This setting is the default in the CLUTO algorithm using cosine similarity as distance measure and was recommended in its manual.
- The top five descriptive terms for each cluster is used to highlight the cluster and used further along in processing to inform sentence selection.

### 3.1.5 Ranking of sentences for each gene

The number of sentences for each gene identified in the 10-year literature corpus ranges from one to 29,203. Users have the option to choose the number of most recent sentences for each gene to be included for processing. The current setting is 100. Sentences are modeled as word vectors after parsing, stop word removal and stemming. Each sentence is assigned a score by linear combination of the following features. This approach follows the framework of Edmundson [10] with modifications customized to the biomedical domain. Sentence score S is calculated as:

$$
\begin{aligned}
S= \quad & w_1\, CluSim + w_2\, QuFreq + w_3\, NGene + w_4\, CTword \\
& + w_5\, TPword + w_6\, L + w_7\, Recency
\end{aligned}
$$

where *CluSim, QuFreq, NGene, CTword, TPword* , *L* and *Recency* are features defined below and $w_{1\text{-}7}$ are weight parameters between 0 and 1 for each feature.

- Cluster representation (*CluSim*). By default, CLUTO gives five descriptive terms to summarize a cluster. It is a good starting point to represent the cluster. Further testing of different size of cluster representation was presented in post-hoc optimization. The top five descriptive features (a set of MeSH, GO terms and/or words) for each gene cluster from the previous step are used as this ranking measure. *CluSim* is calculated as the normalized (against the highest number in the sentence set) number of feature terms the sentence has (for GO and text terms) or assigned to the abstract where the sentence is extracted (for MeSH terms).

- User-enter query terms (*QuFreq*). *QuFreq* is calculated as the normalized (against the highest number in the sentence set) frequency of the query terms (in stemmed form) entered by user.

- Gene relations (*NGene*). Sentences referenced to more than one gene/protein names score higher, otherwise, 0. This is a four-level variable:

1. if the sentence refers to additional gene/protein name (other than the gene being studied) and at least one of the additional genes is from the same cluster as the gene being studied, NGene=1.00;
2. if the sentence refers to additional gene/protein names (other than the gene being studied) and even no additional genes is from the same cluster as the gene being studied, but at least one additional gene is from the input gene set, NGene=0.75;
3. if the sentence refers to additional gene/protein name (other than the gene being studied) and none of the additional genes is from the input gene set, NGene=0.50;
4. if the sentence refers to only the gene being studied, NGene=0.00.

According to OHSU biologists consulted during system implementation, they expressed their interest in sentences that have co-occurring genes from the same cluster and from the input gene set. The scale of this feature is to reflect this preference. Emphasis on relations is also reflected in later features, such as relation words in *TPWord*.

- Cue phrases (*CTword*). This is identical to the Edmundson's Cue feature [10] based on the assumption that the importance of a sentence is represented on the presence of certain cue terms. This is a non-domain specific language feature. For example, the term 'conclusion' may indicate importance. The list of Cue phrases is in Appendix 5.
- Domain specific keywords (*TPword*). Biologically relevant keywords were extracted from the Textpresso [46] ontology's several relation-descriptive sections, i.e. *Action, Consort, Effect, Pathway, Purpose, Physical Association, Regulation. TPword* is calculated as count of keywords in the sentence normalized to between zero and one, with the sentence having the maximum count scoring one.
- Length (*L*). Usually the longer the sentence, the more information it contains. *L* is calculated as a two-level variable: sentences with twenty or more words get the score 1; and sentences with less than twenty words get the score of the fraction |sentence|/20. Note that the average sentence length of the dataset is 23 (see Section 4.1). '20' is chosen as a length factor threshold to discount sentences that are shorter than average.
- *Recency* is calculated as a linear scale for the sentences from one to zero, with the most recent sentence getting the score one, and the oldest sentence score zero.

There are many ways to combine these features by adjusting the weight for each measure. In the development phrase, an even weighting scheme was adopted as default. Then in the evaluation phrase, part of the sentence evaluation data was used to adjust the parameters of the scoring scheme. After evaluation, all the evaluation data can be used to get a better tuning of the feature weights.

### 3.1.6 User interface

The user interface consists of six major web pages. The sample screen shots are in Appendix 3 and they are described in detail below:

1. Input page. The user inputs the gene list for processing here. There are two ways for gene list input. One is to type-in, or copy and paste the gene names and corresponding fold changes to a text field. Down the road, the different fold changes are represented by color-coded presentation of the gene names (up-regulated as red and down-regulated as green.) The other is to upload a tab-delimited text file containing the gene names and fold changes. The advanced options button on this screen takes the user to the option page.
2. Advanced option page. The options allow users to have more control over the features used in the clustering process and sentence ranking. The options are cluster features selection (GO, MeSH and text), MeSH subtree selection, stemming (Y/N), number of sentences included in process.
3. Gene set information page. This is the page next in the process after entering the gene list. This page presents the information of the input list of the genes after the initial processing. This includes the duplicated gene names and fold changes reminding the users of the degenerating nature of the gene name mapping in the system. Next, the gene names that do not map to any MGI_ID are listed and

with links to PubMed for direct search. Finally, all the gene names that map to MGI_Ids are listed with the MGI_Ids and links to the MGI website.

4. The cluster results page. This page provides the clusters and the top five descriptive keywords to the user. The top five terms associated with each gene are also presented. The hyperlink to each gene name leads the user to information for that gene from MGI website. The user can also enter query terms for each gene for the next step: sentence ranking.
5. Summary sentence page. This page presents the sentences as the summary of a gene in a descending ordered form, according to the scores calculated as described in the previous session.
6. Re-arranged sentences page. For the convenience of the user, the sentences are further arranged so that sentences from the same abstract are presented together. The sentence groups are ordered descending by the highest sentence score in the group.

### *3.2 __Evaluation__*

The evaluation methods for summarization can be classified in two categories: intrinsic and extrinsic evaluations [47]. *Intrinsic evaluation* is evaluating the system itself, usually involving calculation of some kind of measurement as compared to a gold standard of correct answers. *Extrinsic evaluation* is to measure how the system helps users in the completion of certain task. In this project, the evaluation used was intrinsic evaluation using expert opinion as gold standard. The experts judged how well the clustering and sentence ranking algorithms work in a task-like situation with the microarray experiment data they were analyzing at the time of evaluation. But since the final result of the task – successfully analysis of the array data completed with the help of the system - was not objectively measured, this is still considered an intrinsic evaluation.

This evaluation study using human subjects was approved by the Institutional Review Board of Oregon Health & Science University, IRB #00003090.

### 3.2.1 Definition of the terminology used in the evaluation process.

The following definitions were given to the evaluation participants to guide their judgment.

*Gene set* – a set of gene names obtained from a microarray experiment. They represent the 'hot list' – genes that are differentially expressed. The size of a gene set ranges from 50 to 100. For evaluation purpose, the participant labeled up to five genes where they had expert knowledge. For example, they may have labeled genes that they are studying or have studied, or have done literature research on.

*Summary terms/descriptive terms* – for each cluster, the clustering algorithm outputs the most common words (default is five words) to describe the cluster. These words were shown to the judges to test if they influenced the subject's judgment of the quality of the clusters.

*Informative term* – if a summary/descriptive term describes certain aspects of the gene, it is an informative term for this gene.

*Meaningful cluster* – if the genes in the cluster are functionally similar or related, the cluster is a meaningful cluster.

*Relevant sentence* – If the sentence provides helpful information for the gene and the gene cluster in the context of this microarray experiment, or the expert believes that after reading the sentence, he/she would like to explore more on the topic and would go to the abstract where the sentence came from, this sentence is a relevant sentence.

### 3.2.2 Evaluating the clustering algorithm.

Five gene sets from the result of five different microarray experiments were tested on the GICSS system by OHSU-based mouse genomic researchers. Each person rated the gene set generated by his or her own lab. For each gene set, the participants labeled the genes they were familiar with. Each of the participants compared cluster pairs, which had at least one of the familiar labeled genes. This setup had the scientists work on the genes they picked to ensure that each person had the expertise for the particular gene to judge the result. First, participants judged the usefulness or meaningfulness of two clusters for each gene set by comparing clusters with random grouping, both including the familiar gene. Then, the effects of different clustering features (MeSH, GO, text) were evaluated by comparing clusters generated by each feature side by side. Finally, the top five summary/descriptive terms for each cluster were shown to the participants and they were asked to change their judgment if needed after seeing the summary terms. For each cluster pair, participants chose the more useful cluster of genes from the pair using a 5-point Likert scale:

1. *cluster on right is absolutely better,*
2. *cluster on right is better,*
3. *they are the same,*
4. *cluster on left is better, and*
5. *cluster on left is absolutely better.*

An option was also offered to allow the participants to indicate that he/she was not able to decide the quality of the cluster pair (*0. I don't have enough information to decide.)* The left/right order of the clusters was randomized in run-time during the evaluation. Table 2 summarizes the cluster pairs evaluated by the participants.

**Table 2. Summary of the cluster evaluation. *Yes* means the pair of cluster features are compared by participant. *No* means no comparison was made for the pair. *Before and after showing keywords* means two comparisons were made by the participant; once before showing the keywords for the clusters, once after.**

| **Cluster Features** | **Random** | **Go terms** | **MeSH headings** | **Text words** |
|---|---|---|---|---|
| **GO terms** | Yes | No | Yes (before and after showing keywords) | Yes(before and after showing keywords) |
| **MeSH headings** | Yes | Yes (before and after showing keywords) | No | Yes(before and after showing keywords) |
| **Text words** | Yes | Yes(before and after showing keywords) | Yes(before and after showing keywords) | No |

The results of the comparison were ordinal data ranged from 1 to 5 with 3 being no preference versus random and zero was treated as missing. They were first normalized to the range of -1 to 1 for each participant, with zero being no preference from random. The transformation was performed as following:

NewScore= (oldScore-3)/Max(|oldScore$_i$-3|) i=1 to 6 within the six judgment scores (three judgment for each of the two genes) by the same participant.

This transformation was used to adjust for the different ranges participants may in for their judgments. The transformed results were analyzed using a general linear model and two-way ANOVA to assess the effects of both the participants and the clustering features.

The results for comparison of rankings before and after showing the cluster keywords were analyzed by two related sample Wilcoxon signed rank test.

### 3.2.3 Evaluating ranking of informative sentences.

Sentences for ten genes (genes from each of the clusters evaluated in the previous step) were used in this step. Sentences from the output of the system and PubMed searches were pooled together and judged by the same scientists who studied the gene set. The searches on PubMed were done by e-search provided by Entrez Programming Utilities. The queries were the name of the gene and synonym expansion using the synonym dictionary from [45]. Once the results were returned, they were filtered on Date of Publication (DP) to limit to the time period of 1994-2003 and on MeSH term (MH) '*Mice*'. These filtering criteria are the same as the text collection, making the comparison between PubMed search results and system output possible. The list of queries for the PubMed searches was in Appendix 6.

For the pooled sentences, the raters assigned an R (relevant) or NR (not relevant) label to each sentence by judging if it had relevant information for understanding the specific gene studied in the microarray experiment they were analyzing. Results from two genes were used to hand-tune the ranking parameters and the other eight were analyzed and used to study the system. Three sentence presentations were compared by average precision (AveP) using the relevance judgments as a gold standard:

1. *GICSS system output*: Sentences with reference to the gene extracted from the abstracts ranked by the scoring algorithm.

2. *Same sentence set as in 1 but in reversed chronological order*, same as PubMed's ranking.
3. *Output from PubMed search* (title of abstract in reversed chronological order).

In IR, precision is calculated as

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

and average precision is $\mathrm{Ave}P = \frac{\sum_{r=1}^{N}(P(r) \times \mathrm{rel}(r))}{\text{number of relevant documents}},$. Mean average precision (MAP) is the mean value of the average precisions computed for each of the queries separately. To evaluate the effectiveness of the three sentence presentations, the AveP and MAP measures were adopted from IR with sentence as the unit of retrieval, instead of document. For each sentence presentation, the AveP scores were calculated for each of the genes and the scores were analyzed using repeated measure with post-hoc comparison with the Sidak adjustment.

Even though AveP score combines both recall (defined in IR as

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

) and precision, its emphasis is recall more. Note that instead of sentences from the abstract, the PubMed output only includes the titles of the abstracts. Because of this, it would likely fare worse by measures that focus on recall. In order to make the comparison more meaningful, precision at 5 (P@5) and precision at 10 (P@10) were also calculated for each sentence presentation set for each gene. Both measures were also analyzed using repeated measure with post-hoc comparison with Sidak adjustment for multiple comparisons.

### 3.2.4 Micro-evaluation of individual features.

Using the relevance judgments 'gold standard' from the previous evaluation step, MAP for sentence ranking using each of the single ranking features  were calculated to study the contribution of each feature. This data was also used to perform further tuning of the ranking feature parameters in post-hoc optimization.

### 3.2.5 Demonstration of generalizability.

To demonstrate the generalizability of the system, a human gene information summarization system was implemented by substituting a gene/protein entity recognition and normalization system for human genes [45].  The goal of this step was to demonstrate the simplicity of adapting the system for different species by simply replacing the NER for different species. The time spent in the implementation was used as measurement for generalizability.

# Chapter 4. Results

## *4.1 Descriptive statistics of sentence database*

In the mice subset of the 1994-2003 MEDLINE records, there are 284,900 abstracts (PMIDs) covering 11,311 mouse genes. The number of sentences in the abstracts ranges from one to 102, with the mean value at 8.36. The length of the sentences in the database ranges from one to 236 words with a mean at 23.39.

There are 583,388 total sentences with reference to at least one mouse gene in the database. The number of sentences for each gene ranges from one to 29,203 (*tumor necrosis factor, MGI:104798).* The mean number of sentences per gene is 99.52.

## *4.2 Gene clustering*

The number of clusters for each gene set depended on the number of genes in the list and the natural diversity of the gene set. Datasets obtained from similar microarray experiment platforms were chosen for evaluation. The size of the gene sets used in the evaluation process ranged from 53 to 275, which we believed represented the numbers of differential expressed genes scientists produced in their real world microarray experiments. However, the criteria to choose these sets of differentially expressed genes were set individually by each scientist for their own data, without any intervention from the author. Therefore, the fold change cutoff values varied. The numbers of clusters generated by the clustering algorithm for each gene set roughly follow the size of the gene set, i.e. the larger the gene set the higher the number of clusters (Table 3). The number of genes in the 10 clusters evaluated by the scientists ranged from four to 12.

General linear two-way (with feature type as the fixed factor and judges as the random factor) ANOVA model analysis indicated no significant difference among the three features (MeSH, GO and text) and judges. Marginal means for each feature showed that both GO and MeSH were significantly better than zero (equivalent to random grouping) with 95% confidence intervals of [0.252, 1.081] and [0.141, 0.970] respectively, while text was not significantly better than zero with its 95% confidence interval range [-0.081, 0.748] (Figure 2). Furthermore, the observed power calculated by the analysis was only 16%, which indicated the analysis may have been somewhat underpowered to detect some potential differences of this magnitude.

**Table 3**. **Descriptive information about the evaluation gene sets. The size of the gene list covered a good range of numbers of differentially expressed genes scientists obtain from microarray experiments. The experiments were performed on the Affymetrix 430A platform, which would ensure the comparability of the data. The criteria for the gene selection were set by individual participants for their own dataset. The number of clusters built by the algorithm for the five gene sets roughly follows the trend of the gene set size.**

| **Gene set** | **Microarray experiment platform** | **Number of genes** | **Gene set selection criteria** | **Number of clusters** |
|---|---:|---:|---:|---:|
| Bagby.txt | Affymetrix 430A array | 74 | >5 fold decrease or > 4 fold increase over wild type | 13 |
| Nikki_rma.txt | Affymetrix 430A array | 275 | q<0.3 (p<0.002) | 27 |
| Dec24hr.txt | Affymetrix Mouse genome 430 2.0 array | 77 | >1.8 fold decrease vs. control | 12 |
| Inc24hr.txt | Affymetrix Mouse genome 430 2.0 array | 78 | >1.8 fold increase vs. control | 12 |
| Inc3hr.txt | Affymetrix Mouse genome 430 2.0 array | 53 | >1.8 fold increase vs. control | 14 |

**Figure 2. The comparison of the three features (Text, MeSH and GO) for their usefulness for building meaningful clusters. Normalized preference at zero indicates no preference versus random grouping, while one indicates absolutely better than random. Clusters generated with MeSH and GO terms were significantly better than random grouping while clusters from Text were not. In addition, the overlapping 95% confidence interval for the three features suggested there was no significant difference for preference among clusters generated by Text, MeSH or GO terms.**

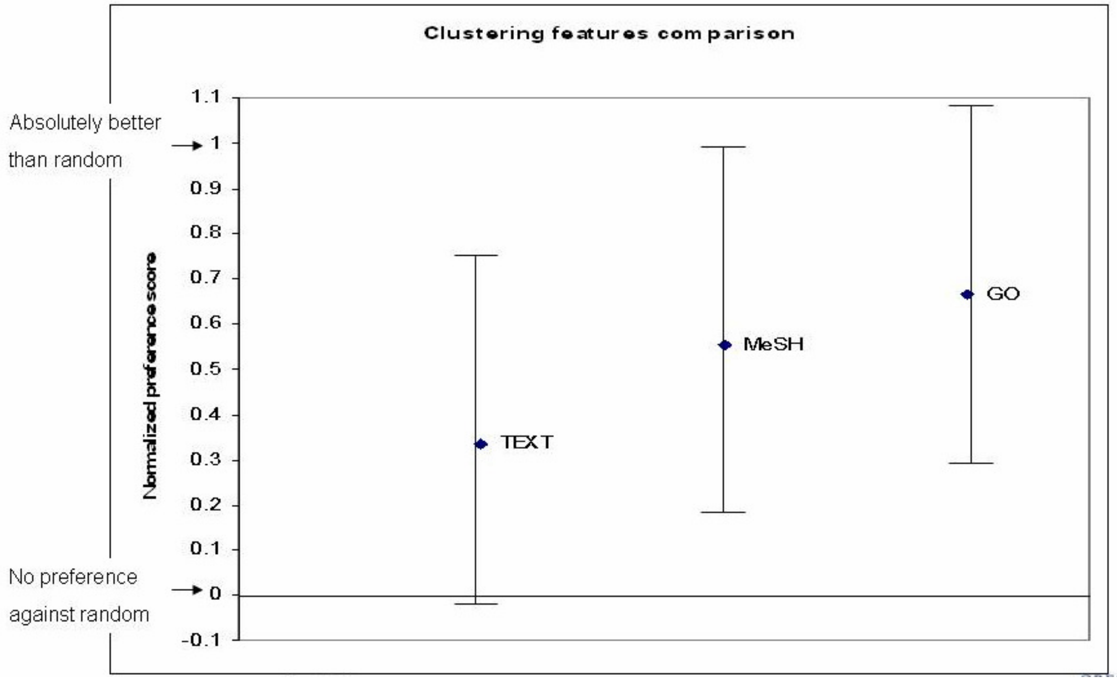
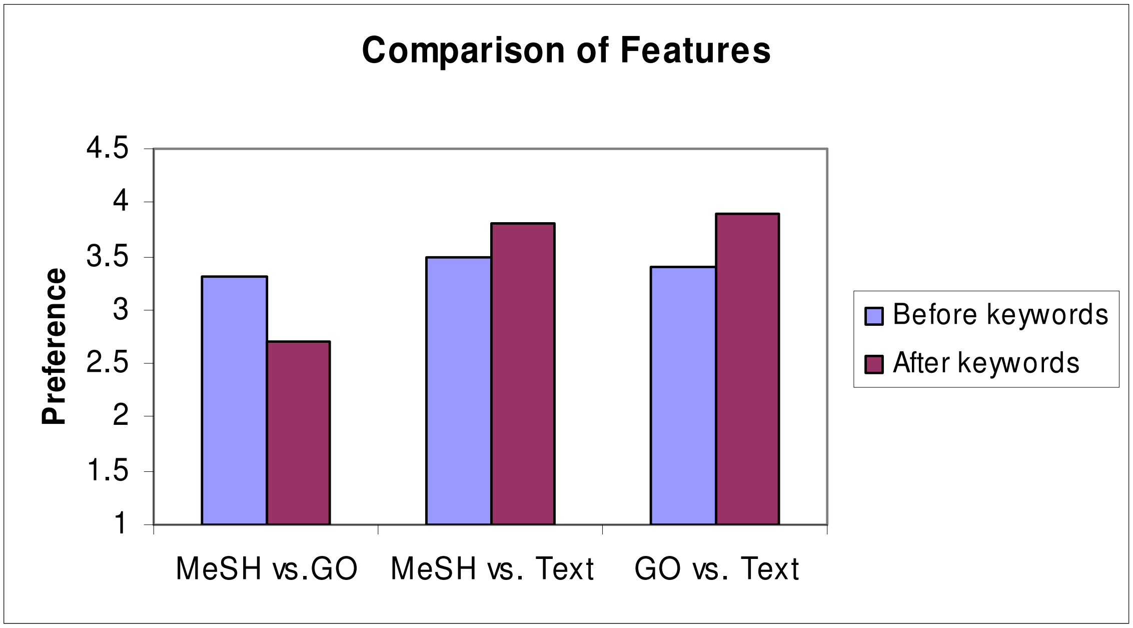


**Figure 3. Side by side comparison of preference for cluster before and after showing the keywords for the clusters to the participants. Preference score=5 indicates participants prefer the left side of the pair (i.e. MeSH in 'MeSH vs. GO' label in the figure), while score=1 indicates participants prefer the right side of the pair (i.e. 'GO') and score=1 means no preference. The data showed that the change of preference after showing the keywords were not significant, suggesting the preference of the participants was not influenced by the keywords.**



To study whether showing the keywords for the clusters influenced the judgment of meaningful clusters, Wilcoxon's signed ranks test was performed on the before-and-after-keyword paired tests. The results indicated that presentation of keywords did not significantly influence the preference of cluster choices for all of the three features. Figure 3 depicts side-by-side the change of preference after showing the keywords for the clusters. While the difference was not significant, the preference scores for GO clusters and MeSH clusters increased over text clusters after showing the keywords.

### *4.3 Sentence ranking*

Recall that the sentence score was calculated as:

$$S = w_1\,CluSim + w_2\,QuFreq + w_3\,NGene + w_4\,CTword + w_5\,TPword + w_6\,L + w_7\,Recency$$

where *CluSim, QuFreq, NGene, CTword, TPword , L* and *Recency* were features defined below and $w_{1\text{-}7}$ are weight parameters between 0 and 1 for each feature. Sentences were ranked by decreasing score.

The parameters for the each of the features were hand-tuned using the results of the first two genes. $w_2$ was set at zero because the use of QuFreq was not evaluated in this evaluation step, but is presented here for model completeness. The rest of the parameters were: $w_1$ =0.1, $w_3$ =0.1, $w_4$=0.2, $w_5$ =0.2, $w_6$=0.2, and $w_7$=0.2.

The number of sentences/titles for each gene studied in the evaluation process ranged from only four to 100 (Table 4). There were more than 100 sentences referencing the gene *cxcl12* in the database and only the most recent 100 were retrieved due to the default value for maximum sentences per gene in the GICSS system. The number of shared abstracts from GICSS and PubMed searches varied depending on the gene. For example, in the case of *usp18,* both GICSS and PubMed searches gave results from the same seven abstracts. One the other hand, results for *irak3, cxcl12 and kcnj9* came from two different sets of abstracts. In general, there were more sentences from GICSS than titles from PubMed. There were also common that more than one sentence from each abstract were returned in GICSS as indicated by the number of abstracts represented in the GICSS output. The number of abstracts represented in the top 10 sentences from the GICSS output ranged from three to eight (Table 5).

The AveP scores for the three sentence presentations for each gene showed that there was some variation in AveP. In addition, AveP scores for PubMed search results were consistently much lower than GICSS output. Furthermore, GICSS output was also consistently better than ranked by recency sentence presentation with only one exception in gene *kcnj9* (Table 6).

**Table 4**. **The number of sentences/titles judged by the participants for each of the genes. In general, there were more sentences for each gene from the GICSS output than the search results from PubMed. The number of shared abstracts by both ouputs varied depending on the gene. The 100 sentences for *cxcl12* may represent the default maximum of sentences output per gene in GICSS.**

| **Gene name** | **GICSS output** | **Number of abstracts represented in the GICSS output** | **PubMed** | **Number of shared abstracts in GICSS and PubMed searches outputs** |
|---|---:|---:|---:|---:|
| adamts1 | 66 | 18 | 17 | 12 |
| usp18 | 38 | 7 | 7 | 7 |
| irak3 | 14 | 4 | 16 | 0 |
| cxcl12 | 100 | 22 | 16 | 0 |
| kcnj9 | 46 | 17 | 60 | 0 |
| pglyrp1 | 71 | 16 | 18 | 16 |
| ptx3 | 90 | 18 | 50 | 13 |
| clca1 | 68 | 29 | 37 | 22 |
| cyp2j5 | 13 | 3 | 4 | 3 |
| frk | 72 | 15 | 11 | 4 |

**Table 5 Coverage of the GICSS output. The number of abstracts represented in the full GICSS output and top 10 sentences is shown. The top 10 sentences from the output have covered a good percentage of the distinct abstracts represented by the full output.**

| Gene name | GICSS output | Number of abstracts represented in the full GICSS output | Number of abstracts represented in the top 10 sentences of the GICSS output | Percentage of abstracts represented in top 10 sentences |
|---|---:|---:|---:|---:|
| adamts1 | 66 | 18 | 3 | 17% |
| usp18 | 38 | 7 | 5 | 71% |
| irak3 | 14 | 4 | 4 | 100% |
| cxcl12 | 100 | 22 | 7 | 32% |
| kcnj9 | 46 | 17 | 7 | 41% |
| pglyrp1 | 71 | 16 | 6 | 38% |
| ptx3 | 90 | 18 | 8 | 44% |
| clca1 | 68 | 29 | 7 | 24% |
| cyp2j5 | 13 | 3 | 3 | 100% |
| frk | 72 | 15 | 6 | 40% |

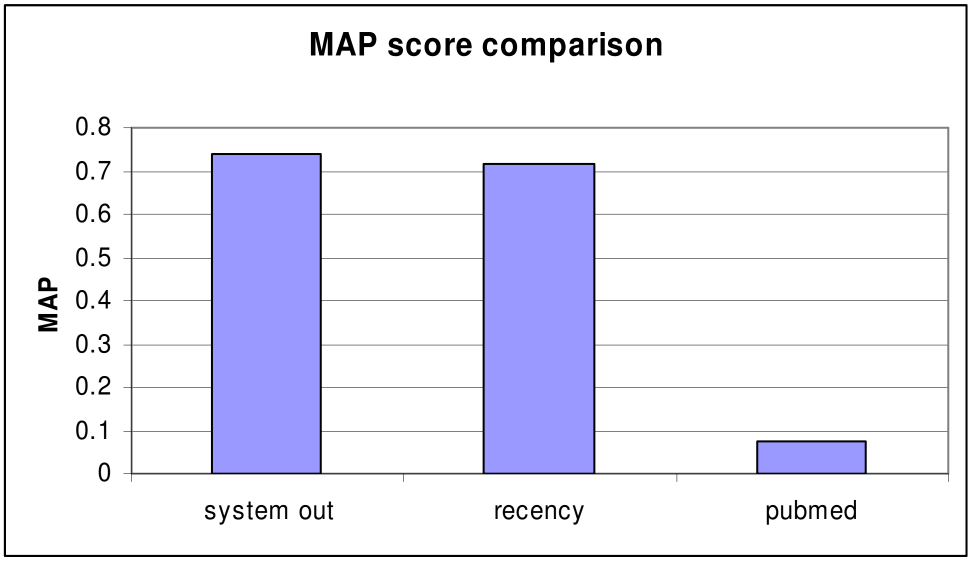
**Table 6. Average precision scores for the three sentence presentation (GICSS output, sentences list produced by ranking with recency and PubMed search output) for each of the eight genes used in this part of evaluation. In general, GICSS output fared better than PubMed search results. GICSS output was also consistently better than ranked by recency presentation with only one exception (*kcnj9*).**

| Gene name | GICSS output | Recency | PubMed |
|---|---|---|---|
| adamts1 | 0.580264 | 0.534627 | 0.039064 |
| usp18 | 0.921985 | 0.900525 | 0.103968 |
| irak3 | 0.654296 | 0.615089 | 0.033333 |
| cxcl12 | 0.854571 | 0.843545 | 0 |
| kcnj9 | 0.647421 | 0.666607 | 0.05862 |
| pglyrp1 | 0.780637 | 0.717311 | 0.117634 |
| ptx3 | 0.79435 | 0.783601 | 0.047008 |
| clca1 | 0.683284 | 0.667058 | 0.205523 |
| **Mean Average Precision** | **0.739601** | **0.716046** | **0.075644** |

A graphical comparison of MAP for the three sentence presentations showed that sentences from the abstract did much better than just titles from PubMed output. The ranking algorithm (as judged by MAP) gave a 3.3% increase over reverse chronological order of sentences and close to ten-fold increase above PubMed titles (Figure 4).

Repeated measure ANOVA analysis of average precision scores for the eight genes suggested that overall there was significant difference among the three sentence presentations ($p < 0.001$). Post-hoc comparison with Sidak adjustment for multiple comparisons indicated that the GICSS system output was significantly better than PubMed ($p < 0.001$). However, the difference between the GICSS system output and sentence presentation produced by ranking with recency was not significant at the 0.05 level ($p=0.08$).

**Figure 4**. **Comparison of the three rankings (GICSS system output, sentence presentation ranked by recency and PubMed search results). It shows that GICSS system achieved a much higher MAP score than PubMed search output (close to 10-fold), while the difference between GICSS system output and sentence presentation ranked by recency was smaller at 3.3%.**

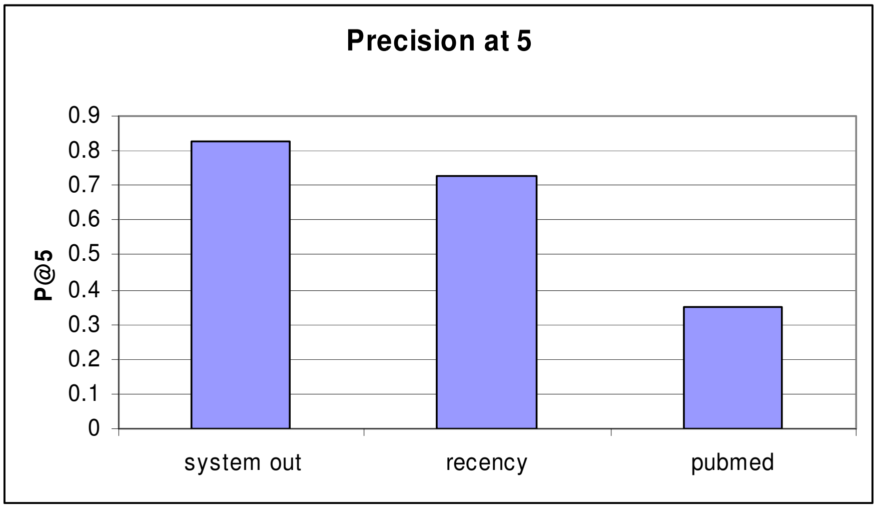


In addition to average precision scores, measurements emphasizing precision were also computed and compared for the three sentence presentations. GICSS output gave 13.8% and 10% increase over ranked-by-recency presentation on P@5 and P@10 respectively. GICSS also outperformed PubMed search output in both P@5 and P@10 (Figure 5).

Repeated measure ANOVA analysis of P@5 and P@10 scores for the eight genes suggested that overall there was significant difference among the three sentence presentations ($p < 0.001$). Post-hoc comparison indicated that the difference between GICSS system output and PubMed was significant ( $p = 0.007$ for P@5 and $p = 0.001$ for P@10), but the difference between GICSS system and ranked-by-recency presentation was not significant at 0.05 level for either of the P@5 ( $p = 0.17$) and P@10 ( $p = 0.11$) measures.

The relation between recency and the probability of being judged as a relevant sentence was further investigated. Judged sentences for all eight genes were pooled and separated into either relevant or not-relevant groups. There were a total of 578 sentences with 357 judged relevant and 221 judged not-relevant. The distribution of both groups over date of publication (DP) was studied. The box plot of the two distributions indicated that the median difference between date of publication (DP) for relevant and not-relevant sentences is close to two years. This indicated that the more recent the sentence, the more likely it is judged as relevant by the participants in this evaluation study. (Figure 6)

**Figure 5. Comparison of mean precision at 5 and precision at 10 scores for the three sentence presentations. The results for P@5 and P@10 were consistent with MAP, with GICSS system output performed significantly better than the PubMed search results while the difference between GICSS system and ranked-by-recency presentation did not reach statistically significance in both of the measurements.**



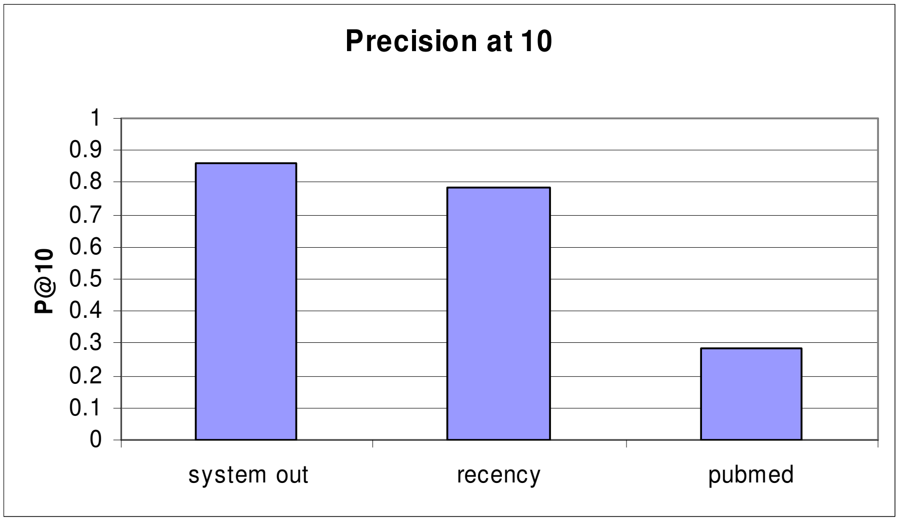
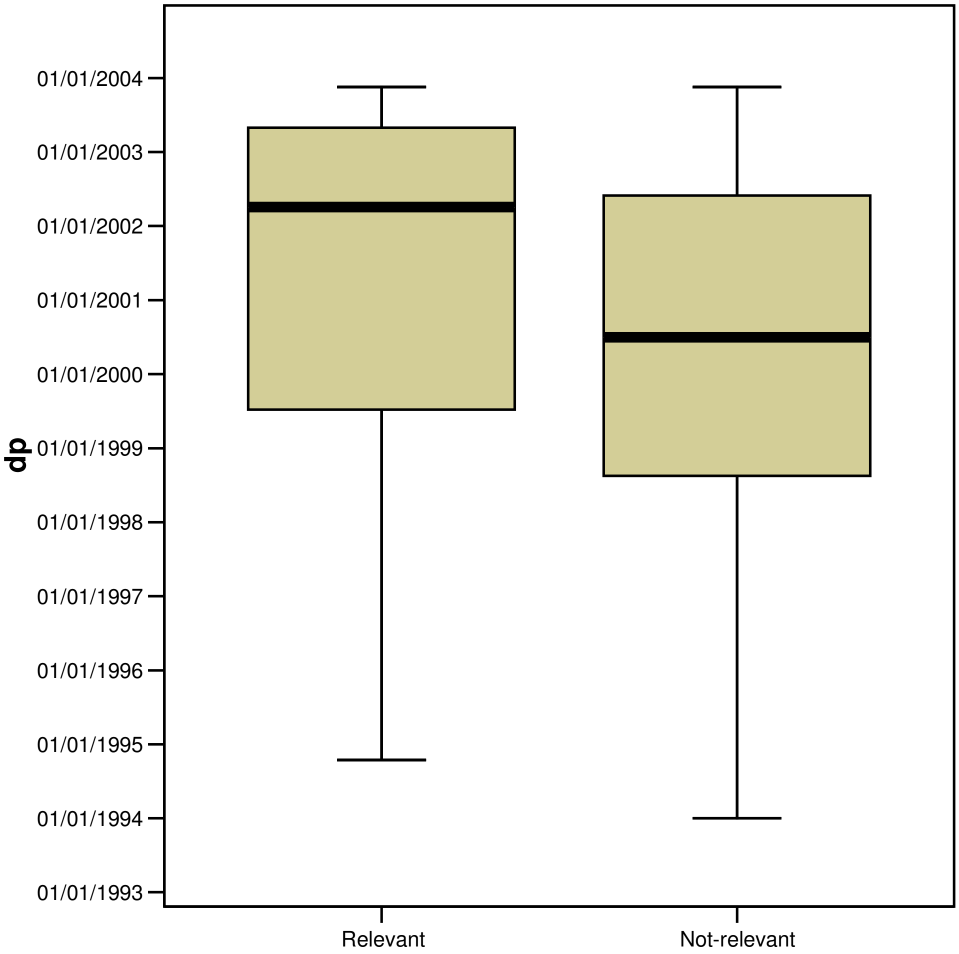


**Figure 6**. **Box plot graph of the distributions of relevant and not-relevant sentences over date of publication (DP). The median for DP for the relevant sentences is in April 2002, while DP for non-relevant sentences is July 2000.**



### *4.4 Individual features' contribution*

MAP scores were also calculated with sentences ordered by each feature individually. The results suggested that domain-specific terminology as represented by the Textpresso protein function ontology *TPword* was the most useful feature and using it alone performed better than the default combination of different features. The feature that represented that context of the cluster, *CluSim* was the least useful features used by itself for sentence ranking and it performed, in the evaluation, worse than random ordering of the sentences (Table 7).

In further experiments we used equal feature weighting in the sentence ranking algorithm and leave-one-out performance (MAP score calculated with the rest of the features when one individual feature is left out) as measure for individual feature performance and obtained similar results (Table 8). Note that in post-hoc studies, by increasing the number of *CluSim* terms, this conclusion was revised.

Since the equal feature weighting scheme performed a little better than the scheme used in evaluation, in post-hoc studies, the equal feature weighting scheme was used as a standard of comparison.

**Table 7**. **Each individual feature's contribution as shown in MAP scores for sentences ordered by each feature alone. It appears that *TPword* was the most useful single feature while *CluSim* was the least useful for sentence ranking.**

| **Outputs** | **MAP** |
|---|---:|
| System | 0.739601 |
| Random | 0.68166 |
| CluSim | 0.675697 |
| Ngene | 0.705838 |
| Ctword | 0.718035 |
| Tpword | 0.75364 |
| Length | 0.726827 |
| Recency | 0.716046 |

**Table 8. Each individual feature's contribution as measured by leave-one-out MAP scores. In the table, the MAP score difference between leave-one-feature-out ranking and the original system (with all features) were displayed for each of the six features. The experiment gave similar results as in Table 6. *TPword* appeared to be the most useful feature – taking it out of the full system lowered the performance by -.023 while *CluSim* was the least useful for sentence ranking - showing improvement of performance if *CluSim* is left out.**

| **Left out feature** | **MAP score with one feature left out** | **MAP score difference between leave-one-feature-out ranking and original system output** |
|---|---|---|
| All features, equal weight | 0.742643 | 0 |
| CluSim | 0.744556 | + 0.001910 |
| Ngene | 0.740432 | - 0.002210 |
| Ctword | 0.737308 | - 0.005334 |
| Tpword | 0.719519 | - 0.023124 |
| Length | 0.733589 | - 0.009054 |
| Recency | 0.720938 | - 0.021705 |

## *4.5 Demonstration of generalizability*

The human gene information summarization system was built by substituting a human gene and protein name entity recognition and normalization system [45]. The system was implemented and became functional after one person/one day's coding and testing. It is currently hosted at http://ir.ohsu.edu/jianji/human_gene. Further analysis of the quality and effectiveness of the human system was not the focus and was not performed for this dissertation.

## *4.6 Post-hoc optimization of sentence selection scheme*

The sentence ranking evaluation results indicated that the *CluSim* feature was not as useful as the other features. With the sentence relevance judgment 'gold standard' obtained from the evaluation process, the sentence ranking scheme was optimized by testing the influence of clustering algorithm on the usefulness of the *CluSim* feature in sentence ranking. Recall that the sentence score was calculated as:

$$S= \quad w_1\, CluSim + w_2\, QuFreq + w_3\, NGene + w_4\, CTword$$
$$+ w_5\, TPword + w_6\, L + w_7\, Recency$$

where $w_{1\text{-}7}$ are weight parameters between 0 and 1 for each feature.

### 4.6.1 *CluSim* feature improvement

By varying the clustering parameter, the following experiments tried to improve the usefulness of the *CluSim* feature contribution and the overall sentence ranking efficiency. The test results suggested that lowering the cluster size (and increasing the number of clusters), i.e., the genes in the clusters became more homogenous, could improve the usefulness of the *CluSim* parameter and improve the overall MAP score. In an experiment where the size of clusters was set to four (note that in original setting the average size of the clusters generated was seven.) *CluSim*'s contribution was higher than *ctword* and *ngene* whereas it was not so effective with the original clustering algorithm settings (Table 9). Note that when the number of cluster parameter was set to a fixed number, CLUTO will attempt to generate clusters of defined size but final size may vary from cluster to cluster, i.e. there may be a cluster of size two, three, four, five or six, depending on the natural property of the gene sets.

Another experiment tested the influence of the size of the C*luSim* features to its contribution in the overall MAP score of the sentence ranking scheme. With all other clustering parameters remained the same as the original setting (cluster number was decided at run-time as explained in section 3.1.4), the size of the descriptive features from each CLUTO cluster was changed from five to ten, fifteen and twenty. The results suggested increasing the feature size improved the C*luSim* feature from *not useful* (Table 8) to a useful effect (Table 10).

**Table 9. Overall system performance and individual features' contribution to the final MAP score when the cluster size parameter for CLUTO was set to four. In the table, the MAP score and the difference between leave-one-feature-out ranking and the full system (with all features) were displayed for each of the six features. All individual features contributed to the final MAP score, with *recency* and *tpword* being the highest contributors and *ngene* and *ctword* the lowest. Decreasing the size of the cluster seemed to generate better *CluSim* features for the summarization process.**

| **Left-out feature** | **MAP score with one feature left out with original settings** | **MAP score difference from the full system with original settings** | **MAP** | **Difference from the full system** |
|---|---:|---:|---:|---:|
| All features, equal weight | 0.742643 | 0 | 0.756748 | 0 |
| **CluSim** | **0.744556** | **+ 0.001910** | **0.744556** | **-0.012192** |
| Ngene | 0.740432 | - 0.002210 | 0.752581 | -0.004167 |
| Ctword | 0.737308 | - 0.005334 | 0.750881 | -0.005867 |
| Tpword | 0.719519 | - 0.023124 | 0.736845 | -0.019903 |
| Length | 0.733589 | - 0.009054 | 0.745768 | -0.010980 |
| Recency | 0.720938 | - 0.021705 | 0.736575 | -0.020173 |

**Table 10. Overall system performance and individual features' contribution to the final MAP score with the size of the descriptive features in the *CluSim* feature in the sentence ranking scheme set to 10, 15 and 20. In the table, the MAP score and the difference between leave-one-feature-out ranking and the full system (with all features) were displayed for each of the six features. Increasing the size of the *CluSim* feature seemed to improve its contribution for the summarization process comparing to the original setting of size five.**

| **Outputs** | **Size of the *CluSim* feature =5** | | **Size of the *CluSim* feature =10** | | **Size of the *CluSim* feature =15** | | **Size of the *CluSim* feature =20** | |
|---|---|---|---|---|---|---|---|---|
| | **MAP** | **Difference from the full system** | **MAP** | **Difference from the full system** | **MAP** | **Differenc e from the full system** | **MAP** | **Difference from the full system** |
| System | 0.74264 | 0 | 0.75490 | 0 | 0.75212 | 0 | 0.75517 | 0 |
| **CluSim** | **0.74456** | **+ 0.00191** | **0.74451** | **-0.01039** | **0.74464** | **-0.00749** | **0.74464** | **-0.01053** |
| Ngene | 0.74043 | - 0.00221 | 0.75557 | -0.00067 | 0.75066 | -0.00147 | 0.75375 | -0.00141 |
| Ctword | 0.73731 | - 0.00533 | 0.75108 | -0.00381 | 0.74622 | -0.00591 | 0.74874 | -0.00643 |
| Tpword | 0.71952 | - 0.02312 | 0.72786 | -0.02704 | 0.72202 | -0.03010 | 0.72723 | -0.02793 |
| Length | 0.73359 | - 0.00905 | 0.74956 | -0.00533 | 0.74393 | -0.00819 | 0.74711 | -0.00805 |
| Recency | 0.72094 | - 0.02171 | 0.73872 | -0.01618 | 0.73600 | -0.01612 | 0.73800 | -0.01717 |

### 4.6.2 Effect of length factor threshold

The effect of length factor threshold (originally set to 20) to discount longer sentences was also tested by varying the value to 10 and 30. The clustering parameters were kept in the original setting, i.e., real-time generation of clusters, ***CluSim*** feature size set to five. The result indicated that length factor threshold did not seem to affect the usefulness of the *length* feature, since for the values of 10, 20 and 30, the contribution of length in the sentence ranking algorithm were about the same at 0.009 (Tables 8 and 11).

**Table 11. Overall system performance and individual features' contribution to the final MAP score with the value of sentence length factor threshold set to 10 and 30. In the table, the MAP score and the difference between leave-one-feature-out ranking and the full system (with all features) were displayed for each of the six features. Varying the value of the threshold seemed to have no effect on its contribution for the summarization process.**

| **Outputs** | **Length factor threshold =10** | | **Length factor threshold =20** | | **Length factor threshold =30** | |
|---|---:|---:|---:|---:|---:|---:|
| | **MAP** | **Difference from the full system** | **MAP** | **Difference from the full system** | **MAP** | **Difference from the full system** |
| System | 0.74327 | 0 | 0.74264 | 0 | 0.74246 | 0 |
| CluSim | 0.74835 | +0.00508 | 0.74456 | + 0.00191 | 0.74316 | +0.00071 |
| Ngene | 0.74225 | -0.00101 | 0.74043 | - 0.00221 | 0.73919 | -0.00326 |
| Ctword | 0.74450 | +0.00123 | 0.73731 | - 0.00533 | 0.73967 | -0.00279 |
| Tpword | 0.72130 | -0.02197 | 0.71952 | - 0.02312 | 0.71722 | -0.02523 |
| **Length** | **0.73359** | **-0.00968** | **0.73359** | **- 0.00905** | **0.73359** | **-0.00887** |
| Recency | 0.72748 | -0.01579 | 0.72094 | - 0.02171 | 0.72043 | -0.02203 |

4.6.3 Optimization of the sentence scoring scheme

Final full post-hoc optimization of MAP was done using the Constrained Optimization By Linear Approximation (Cobyla) interface module in scipy[18]. All the system settings remain the same as in the original evaluation except that the size of ***CluSim*** features =10 because this level gave the best result.

Again, the sentence score was calculated as:

$$S= \quad w_1\, CluSim + w_2\, QuFreq + w_3\, NGene + w_4\, CTword$$
$$+ w_5\, TPword + w_6\, L + w_7\, Recency$$

where $w_{1\text{-}7}$ are weight parameters between 0 and 1 for each feature.

$W_2$ is set to zero again since this feature was not used and it is presented in the formula only for model completeness. The optimization was performed on the other six parameters. Eight constraints were passed to the algorithm:

$w_i$ >=0, i=1, 3, 4, 5, 6, 7

Σ $w_i$ -1 >=0, i=1, 3, 4, 5, 6, 7, and

1- Σ $w_i$ >=0, i=1, 3, 4, 5, 6, 7

There were several local optimal points depending on the initial starting point. With the initial guess for the sentence ranking parameter vector set at equal weight for each feature, i.e. [$w_i$, i=1, 3, 4, 5, 6, 7] = [0.166, 0.166, 0.166, 0.166, 0.166, 0.166], and after 81 iterations, the optimization process converged to MAP = 0.7654 with feature weight

[18] http://www.scipy.org/doc/api_docs/scipy.optimize.cobyla.html

vector [$w_i$, i=1, 3, 4, 5, 6, 7] = [0.223, 0.089, 0.004, 0.484, 0.004, 0.195]. In this optimized setting, the order of feature importance is *tpword,* ***CluSim****, recency*, ngene, *ctword* and *length*. Using the rounded-up feature weight scheme, [$w_i$, i=1, 3, 4, 5, 6, 7] = [0.2, 0.1, 0.0, 0.5, 0.0, 0.2], the system performance was at MAP = 0.7651. This MAP score is only a little higher than before optimization with equal feature weighting scheme.

# Chapter 5. Discussion

## *5.1 Functional gene clustering*

In general, the clustering algorithm gave better gene groups than random as supported by the fact that clusters generated by both MeSH and GO terms were considered significantly better than random grouping of genes. The comparison between the feature types showed insignificant differences; even though the confidence interval ranges and trend suggested that MeSH and GO may be better than text as features for clustering. The observed power of the test indicated that the sample size of 10 genes may be too small to give enough power to distinguish between the different features. Future work should include more biologists for testing, preferably during their real use of the system because the experimental setting of test had some limitations which are discussed next. In addition, the system allowed for any combination of features to be used for clustering, but how different clustering feature combinations fare against the single feature types used in this study was not studied here and remains for future work.

The result of the paired testing of before and after showing keywords of the clusters indicated that the preference of the participants was not influenced by the keywords significantly. It appeared that the perception of a good cluster did not depend on the scientist knowing the clusters descriptive terms significantly. Once the participants found the meaningful cluster, they were likely to stick with it even after seeing the keywords. While the difference was small, the participants had a slight preference in GO or MeSH clusters over text clusters after they saw the keywords. This was consistent with the findings in the first part of the evaluation. It suggests that controlled vocabularies fare better than text words for generating clusters. Controlled vocabularies usually have more domain-specific content, which may be able to give more information to users.

The clustering algorithm, direct k-means, used in this project was from the CLUTO clustering suite. The number of clusters (N) is one parameter that needed to be specified for the algorithm. The selection of N is difficult because it highly influences the number of genes in each cluster and the quality of the cluster. Since the number of input gene list genes covers a fairly wide range, the selection of N using the relative change of internal similarity seemed a better choice than presetting a defined number. But this selection scheme was probably not optimal. The post-hoc experiment setting the cluster size to a defined value seemed to improve the *CluSim* feature's contribution, to the extent that a poor *CluSim* feature became a helpful one for sentence ranking. Even though this may not necessary be interpreted as improvement of cluster quality, it is an indication that the cluster keywords represent the genes in the cluster better, and were more useful in selecting good quality sentences. The proper way to determine N given the size of the gene list and the quality of the clusters is another area of further work. The determination of N can also be done by other algorithms that specialized in finding number of classes in a large dataset such as AUTOCLASS as used in [31]. Other clustering algorithms such as ANT [48] that do not require the input of N are another area to investigate.

During the cluster-quality-judging experiment, we found that judging cluster pairs was not an easy task for the scientists. Even though each cluster had at least one of the genes they chose as familiar, it was very common that some genes in the cluster (with average size of 8 genes) were not familiar to them. In order to judge the quality of the cluster, they need to follow links in the evaluation screen for information on other genes in the cluster. This created a larger than expected work load for the evaluators and by the end of the session, they make their best judgments without going through the information for not-so-familiar genes, possibly due to fatigue. This was also the reason each participant was asked to judge clusters on only two genes, which already amounted to 12 cluster pairs in our experimental design. The fatigue factor may also have influenced the quality of the judgment. In order to overcome the difficulty of getting experiment participants and ensure the quality of the judgment, it will be better to conduct future evaluation experiments in a real-world use setting, similar to extrinsic evaluation, instead of system evaluation such as we have done here. For example, when the researchers are using the system to help them do research on a microarray result gene set during the normal course of their workflows, the system could log the clicking and timing of the participants in addition to the defined questions presented to the participants.

How to best measure the quality of clusters is still in general an issue, especially in this case, where we define quality as how meaningful the clusters were for a specific microarray experiment. Some analytical measures, such as internal and external similarities, entropy and mutual information, may not correlate closely. These measures are commonly used to quantify the quality of the clusters in many comparative experiments [36]. No work so far has been done on how the purity of the cluster as defined by these statistical measures correlates with the biological meaning of the clusters for a user. Furthermore, in this study, the raw gene list from the experiment was clustered. By nature, the gene list contains genes that were differentially expressed and most likely from many different pathways and groups especially considering many genes may perform multiple functions.  We expect the list to be harder to cluster than choosing several distinct known gene groups, such as GO and cell cycle groups and try to cluster them to the right class, which is the most common evaluation method used [31,36,37].

The GICSS system takes in differentially expressed gene name lists input by scientists with no pre-defined criteria for 'differentiated expressed' such as a certain FDR cutoff level. As indicated in the evaluation gene sets, the criteria for the selection of the gene set varied (Table 3). If the input gene set was selected by a loose criterion, it is likely the some genes from pathways other than those that are modified by the testing biological conditions would be included. The resulting clusters generated by the algorithm with these 'contaminated genes' will more likely to be less favorable.

### *5.2 Sentence extraction summarization*

Providing sentences in the abstract gave much more relevant information than titles. All three measures (average precision, P@5 and P@10) showed significant differences between the system output and the PubMed search output. These results suggested it may be useful for the PubMed result list to include highlighted keywords from the sentences in the abstracts to provide more information to the searchers. In the current title-only list, some relevant articles may be missed because the titles do not provide enough information to warrant further exploration of the abstract, especially when the returned list is long.

The GICSS system output using the scoring algorithm in point estimate consistently gave higher MAP, mean P@5 and P@10 scores than presentation of the same sentence set by reverse chronological order, even though the difference is not significant at 0.05 level in post-hoc analysis with multiple comparison adjustment. Further analysis of the relation between recency and relevance indicated that the more recent the date of publication, the more likely the sentence be judged as relevant. This is a little surprising because the intention to include recency in the scoring algorithm was mainly to provide the most recent information first and relevance was not the initial consideration. This relation may be explained by the fact that the experts doing the relevance judgment were to a certain extent aware of the knowledge accumulation timeline of the specific genes. It seems that this is a good assumption because the experts were instructed to select two genes they were familiar with to perform the evaluation. Because of their pre-existing knowledge of the genes, they were likely to pick the newly discovered information as more relevant than the well-known facts on the genes.

Domain specific ontology terms as in Textpresso ontology improved results consistently as indicated by this being the highest single feature MAP and highest contributor in all different setting schemes we have tried. In addition, it ranked the first in the final optimization process with a optimal weight at 0.5. It suggested that the domain specific terms can be very useful to identify important sentences. In addition, it may be used in the clustering process as a domain specific term database to filter text terms to improve on the effectiveness of clustering with free text features. With the domain specific term filter, the free text terms can be better representing the biology content of the abstracts discussing the gene. More on this issue is discussed in the Future Work section.

While the domain-specific terminology provides a good indication of sentence importance, the non-domain specific features such as cue words and length of the sentences might also be helpful in selection of relevant sentences. Even though the final optimization indicated the contribution of these two features were minimal, the other experimental results suggested that they did contribute to a certain degree. Also note that the optimization presented was one of the local maximums while other maximums specified higher weights for cue term and sentence length. The fact that slightly different weighting scheme gave similar local maximums suggested that the features used in the sentence scoring algorithm were providing redundant information and may be correlated. Even though it was inconclusive that the non-domain specific features were helpful in this study in the biomedical domain, including cue terms/phrases and length to help the construction of summaries has been used widely in automatic summarization works of the news and other domain articles[10,49]. This has not been used much in the summarization of biomedical domain, such as [42,50]. Due to the limitation of time and resources of this dissertation, the cue word list was assembled by the author (who has a biological background) after reading the top one hundred PubMed abstracts returned with the search term '*gene*'. Further expansion of the list by other domain experts would be very valuable. It is also interesting to study if the cue words in biomedical domain differ from other domain. To the author's knowledge there have not been any studies on the topic.

In the original implementation, the size of keywords for the cluster (*CluSim*) was set to five and the results of the evaluation seemed to indicate that it did not seem to help much. In the original implementation, inclusion of this feature lowered the MAP score of the system output comparing to scheme that left this feature out. *CluSim* is the feature we would like to use to include the context for the sentence selection. It usually represents the overall information for the whole cluster instead of specific information in that particular gene. Therefore, one possible reason that *CluSim* was not so useful in the original implementation may be that the users prefer specific information represented by the ontology terms rather than the general knowledge about the gene group functions. In order to test if different cluster parameters and more descriptive keywords may be able to better capture the cluster information, we varied the cluster size and the number of keywords used in the *CluSim* feature in the post-hoc experiments to test the effect of their influence to the sentence ranking scheme. The results seemed to suggest that feature terms from more homogenous clusters as well as larger feature sets could improve the contribution of the *CluSim* feature in the sentence ranking scheme. In fact this change made it a positive feature, and the COBLYA optimization showed it to be the second best feature. One explanation can be that the *CluSim* terms (descriptive terms for clusters) are mostly domain specific terms as in GO and MeSH. Therefore, increasing the number of the *CluSim* words increases the domain specific feature weighting, i.e., similar to the TPword feature. On the other hand, this assumption did not explain why decreasing the cluster size (more homogenous cluster) can improve the contribution of *CluSim* feature. Furthermore, it did not explain the fact that using a smaller size feature set rendered the feature to negative contributor. Therefore, a more likely explanation is that the quality of the clusters and the richness of the descriptive features for each cluster are important for the usefulness of the *CluSim* feature.

The GICSS system supports the use of query term in selecting the important sentences. In this way, it gives the users more leverage in getting the information of interest. For example, they can enter a disease's name in order to retrieve sentences referring to the gene and the disease. Hopefully, this feature can provide customized sentences presentation to fit different needs of the users. Due to the limited resources and increased need for user initiative this feature was not evaluated in the study.

In the post-hoc optimization, the final optimized MAP score was only a little higher than MAP achieved by equal weighting scheme before optimization. This may indicated that the features used in scoring were unable to distinguish relevant sentences. Testing more features that could be as useful as the *TPword* is part of future work.

The evaluated version of GICSS was tuned by the two sets of sentences. Since there are six parameters and use only two examples for adjustment, there were obviously not enough data to fully tune the system. The results of post-hoc experiments suggested that the system could achieve better performance when tuned by all the data points collected in the evaluation. On the other hand, data from only ten genes may be still too little to fully tune the system. If data can be collected during the real use of the system, it will be very useful in choosing a better weighting scheme. Therefore, distributing of the system for the real users is of high importance once all the evaluation and testing is done.

The system is built on top of a gene name recognition system (NER), the accuracy of which influences the result of both clustering and sentence selection. Our NER achieves state-of-the-art accuracy at 70-80% [51], but the 20% error rate can cause some mistakes cascading down to sentence extraction. For example, two genes (20% of the tested genes) got low MAP scores. One of them was because the sentences were in fact about other genes with identical symbols. The other is because the gene symbol for the gene has a synonym identical to a DNA motif. A module to deal with this type of error can be added to the preprocessing step to disambiguate the gene/protein names in the sentences. This would help to avoid the problem of sentences that have information on totally different genes being mixed with sentences of the gene of interest and therefore achieving better performance.

The NER system does not make any distinction on whether the entity mentioned is protein or gene. In fact, it is very difficult to distinguish between these two types of entities, since the use of gene name and its protein products is mostly interchangeable in the literature, and in many cases can only be inferred from the surrounding context. Is it important to separate the sentences on a gene and the sentences on its protein product? It might be important because the system is focusing on analysis of microarray experiment data and the scientists usually are more interested in information on genes instead of proteins. On the other hand, the information of both gene and the protein gene product may be of great interest because proteins are the entities that influence the physiological activity. The GICSS system currently cannot specify the sentences on genes only, but presents sentences for both gene and protein. At the present time it is unclear whether this distinction is important for systems such as GICSS. Supporting this distinction will rely on the development of better NER systems.

# Chapter 6. Limitation and Future Work

There were limitations of this study but also avenues for future work. In order for the GICSS system to be of real use for the genomic research community, it is important to make the system available to researchers by integrating it into the current microarray analysis workflow. As far as improvements on the algorithmic aspects of the system, further directions include the following.

First, the text collection for this study is a snapshot of the MEDLINE records. In this study, we use a static collection of MEDLINE records from 1994 to 2003. An obvious limitation to this approach is that it does not have the data of the rapid advent of the field in more recent years. This can be easily remedied by downloading MEDLINE records from the NLM website. After these development and evaluation processes, the dataset will be updated automatically each week to include the current data from MEDLINE. Second, it contains only abstracts, while more information is in the full text of the articles. Especially if we try to analyze at transcript level, the information is more likely in the full text instead of the abstract. Even though full text articles are still not easily accessible currently, we expected the increase in publicly available electronic texts. This assumption is supported by the open access movement resulting in not only Biomed Central and PubMed Central, but even for commercial publishers, the trend is that more journals are offering their subscribers and even the public the electronic version of their publications. Another direction for future research is to incorporate available full text in the system and compare with abstract only system. The full text can give us more information, hence providing more details that an abstract can never cover. On the other hand, with full text, certainty of the meaning of the sentences may be an issue, since some sections of the full text such as discussion may include speculation and hypotheses.

Further work can also be done to improve the effectiveness of the clustering algorithm. The CLUTO clustering suite has several different types of algorithms that can be plugged in to test the effectiveness of clustering. In addition, the GICSS system uses the cosine distance as a measure of similarity. This did not take into account the concept structure of the terms used as features for clustering. This semantic relation represented in the hierarchical structures of domain terminologies could be used to improve to quality of clustering. Future work should include testing on the use of semantic distance as similarity measurements, especially for GO and MeSH terms, which have hierarchical structures, possibly to achieve better performance [52].

The text feature for clustering did not perform as well as the other two features. Note that the text feature was processed only by stop-word removal, stemming and tfidf weighting. In order to make better use of the free text feature, further filtering and different weighting of the text content by its biological information density should be helpful. Some of the prior work in this area includes Andrade's study on way to automatic extraction of keywords from text articles [53] and Liu's study on comparison of two feature extraction and weighting schemes[36]. We can adapt their approaches of giving different weighs to different text terms to improve the text feature gene functional clustering results.

The GO terms used in the clustering process were pooled together disregarding the confidence of the assignment. Future work includes working on possible utilizing the evidence codes for each term to assign different weighs for the terms.

Another area of future work is to test the use of NLP techniques. GICSS does not use much advanced Natural Language Processing (NLP) techniques, such as part of speech tagging, semantic parser, word sense disambiguation, co-reference and anaphora resolution. There were both advantages and disadvantages for not using NLP. The most notable advantage is speed. The system can process gene list containing up to 300 genes smoothly (around three minutes), while some of the system using NLP in real time were slow when processing less than 50 gene names [50,54] (personal experience with the website versions in December, 2006: [50] took over ten minutes and [54] never gave results.). NLP can be very useful in extracting predefined information in templates. But it is also limited by the predefined template. The GICSS system can extract useful information in general with no restriction to certain functional types and language templates. Finally, the updating of the templates in NLP is usually time-consuming and might also be influenced by different writing styles of the authors. On the other hand, the disadvantages of using word modeling instead of NLP include that it is not possible to deeply understand the meaning of the sentences. There is no detailed knowledge, such as the direction of the relation -- *protein A activates the expression of gene B* ; cause and effect relation – *disease C is caused by the defects in gene D*. The information presented by the system is more generic but not as detailed as processing by NLP may possibly provides. With NLP’s advantages and disadvantages discussed, it will be interesting to study if adding these features will improve the performance and how to balance the improvement of performance if there is any, with the sacrifice of speed.

The type of NLP used in GICSS is name entity recognition. Gene and protein name entity recognition, synonym detection and normalization are important first steps in the process, on which the system's accuracy depends heavily. It has been noted that gene name ambiguity is low (5%) intra-species and high (85%) inter-species [45]. It makes sense to limit the study to mouse, since NER system for mouse achieves the highest accuracy of all the vertebrate species. But the decision to limit the system to one species also limits the potential application of the system. In addition, articles on the orthological gene in other species (especially close related species, e.g. human and mouse) will potentially provide significant information for the scientists. The GICSS system currently does not support searching of sentences regarding other species. One way to remedy is mapping to orthological genes from the different species and retrieves all available sentences. The drawback is that it will cause higher level of false positive hits due to the error in NER in other species and the added source of error in mapping. The effectiveness of this approach can be a direction in future work.

In the dataset used in the project, the number of publication on genes varies greatly from gene to gene. As mentioned before, the number of sentences for the genes in the 10-year MEDLINE abstract on mouse is ranging from one to 29,203. In general, the highly different level of publication related to different genes reflects the fact that some genes are well-studied and some are barely worked on. For highly studied genes the feature sets are much richer than the genes that have very few abstracts/sentences published on them.

This fact may very likely influence the result of the clustering process, resulting in clusters skewed by the level of publication. The same problem exists for the combination of the three features (text, MeSH and GO) too. The number of GO terms assigned to each gene is usually much fewer than the text word from the sentences. When combining these features, how to compensate for the impact of richness of the feature is another future research direction.

Furthermore, the genomic research technologies are advancing at a fast pace. This trend is reflected in the publication by the appearance of articles describing the results from the experiments using these technologies. The text processing and understanding community working in biomedical domain is always trying to catch up with the need of the experimental community, but still lagging behind. For example, microarray can now measure expression level at the granularity of transcripts. Since the same gene region can be transcribed into different splices (transcripts of mRNA) in different cell condition, or different time point in development, this finer granularity gives scientists more insight into the mechanisms of the gene functions. At present, not much literature mentions transcripts, at least not in the abstract. Hence, the language processing community has not yet been focusing at this level of granularity for name entity recognition, but as the literature on the different transcripts from the same gene accumulates in the future, NER for transcripts may be next area of research. Based on the currently available NER applications, the GICSS system is only able to work at the gene entity level but not as fine as transcript level. Thus, different transcripts from the same gene are forced to map to the same gene name during the input step to the system. Since the GICSS system aims at helping microarray analysis, this limitation potentially has the following two scenarios:

1. Different transcripts of the same gene have the same type of regulation pattern (up/down regulation) and show in the input list as duplication of gene names with different fold change.
2. The different transcripts have opposite type of regulation pattern, i.e. one up and one down regulation, and show in the input list as contradiction, which also leads to the problem of how to color-code that gene and difficulties in interpretation.

Even though we do not have the capacity to analyze our text collection at transcript level, in this version, the GICSS system implements the feature to highlight these duplication and contradiction for users and remind them the degenerative nature of mapping transcripts to gene name. In future work, incorporating the capacity to handle text at gene transcript level will be one of our directions. In order for text analysis to achieve at this finer level, large amount of publication describing experiment results in transcript level is required. Furthermore, NER systems that aim at recognizing transcript names will be essential too. We would work on identifying transcript names in free text and the mapping between gene names and available transcripts. Once the literature reaches the significant threshold, we will be ready to analyze and conduct further studies.

Finally, in the user-interface aspect, there are areas that can be tested in order to give the users more options to adjust the system to their use. Since GICSS is aimed to help the microarray scientists explore their gene sets obtained from the experiments, it would be to the advantage of the users that in addition to one predefined default setting, the system allows them to adjust some of the parameters, such as, the number of clusters, weigh of sentence scoring algorithm to return different clusters. This way, they can explore the difference of the settings and decide which setting is best for the gene set. This notion of allowing users to set some of the parameters is common in the statistic analysis software available for the gene expression profile analysis of the micro-array data. Some of the options, such as selection of features, stemming and number of sentences were available in the current system. This could be a valuable area for future work.

# Chapter 7. Conclusions

A gene information summarization system was built and evaluated for mouse genome researchers working with a large gene list micro-array experiment results. This system, GICSS, aimed to fit between the expression pattern analysis and finding supporting evidence from the literature.

The results of the evaluation indicate that the functional gene clustering approach can generate meaningful gene clusters using both MeSH and GO term features. There is no significant difference between the use of the different features (MeSH, GO and TEXT) as identified by this evaluation, even though the evaluators seemed to prefer clusters built on GO and MeSH features over text features.

By presenting the sentences from the abstract, the system can provide more relevant and important information to the users than standard methods such as PubMed search output with titles ranked by reverse chronological order. This result suggested that an uninformative title in the search result may have the user miss important information in the abstract. Both domain specific ontology terms and non domain specific general language features are useful in the selection of important sentences.

The system used unsupervised learning that does not depend on predefined categories and requires no training data, which is usually hard to get and requires retraining over time. This may make the system more generalizable. The generalizability of the system to other species was also demonstrated by the implementation of human gene information system. In the evaluation, we tried to mimic the real life microarray data analysis situation to measure the true usefulness of the clustering algorithm as judged by the participants working with their own data, as versus to other popular internal hard measures of cluster similarity or the effectiveness of classifying predefined distinct gene groups.

Future work will include distributing the system to generate real use by the scientists in genomic research while working on improvements to the algorithm and automating database updates.

# References

1. Shatkay H, Feldman R. 2003; Mining the biomedical literature in the genomic era: an overview. J Comput Biol 10(6):821-855.

2. Lockhart Dea. 1996; Expression monitoring by hybridization to high-density oligonucleotde arrays. Nat Biotechnol 14:1675-1680.

3. Fattore M, Arrigo P. 2004; Topical clustering of biomedical abstract by self organizing maps. In: Proceedings of 'The Fourth International Conference on Bioinformatics of Genome Regulation and Structure'; Novosibirsk, Russia, July 25-30, 2004.

4. Hersh W, et al. 2005; TREC 2005 Genomics Track overview. The Fourteenth Text Retrieval Conference, TREC 2005.

5. Rzhetsky A, Iossifov I, Koike T, Krauthammer M, Kra P, Morris M et al. 2004; GeneWays: a system for extracting, analyzing, visualizing, and integrating molecular pathway data. Journal of Biomedical Informatics 37(1):43-53.

6. Novichkova S, Egorov S, Daraselia N. 2003; MedScan, a natural language processing engine for MEDLINE abstracts. Bioinformatics 19:1699-1706.

7. Hersh W, et al. 2006; TREC 2006 Genomics Track overview. The Fifteenth Text Retrieval Conference, TREC 2006.

8. Sparch Jones K. Automatic summarizing: factors and directions. In: Mani I, Maybury MT, editors. *Advances in Automatic Text Summarization.* London: MIT Press, 1999.

9. Luhn H. 1958; The Automatic Creation of Literature Abstracts. IBM Journal:159-165.

10. Edmundson H. 1969; New methods in automatic extracting. Journal of the ACM 16(2):264-285.

11. Radev D, Jing H, Stys M, Tam D. 2004; Centroid-based summarization of multiple documents. Information Processing and Management 40:919-938.

12. Teufel S, Moens M. 2002; Summarizing Scientific Articles: Experiments with Relevance and Rhetorical Status. Computational Linguistics 28(4):409-445.

13. Mani I. Automatic summarization. Amsterdam, Philadelphia: J. Benjamins Pub. Co., 2001.

14. Kupiec J, Pedersen J, Chen F. 1995; A trainable document summarizer. Proceedings of the 18th annual ACM SIGIR conference on Research and development in information retrieval:68-73.

15. Nomoto T, Matsumoto Y. 2001; A new approach to unsupervised text summarization. Proceedings of the 24th annual ACM SIGIR conference on Research and development in information retrieval:24-36.

16. Conroy JM, O'Leary DP. 2001; Text summarization via hidden markov models. Proceedings of the 24th annual ACM SIGIR conference on Research and development in information retrieval:406-407.

17. Marcu D. 1996; Building up rhetorical structure trees. American Association for Artificial Intelligence 2:1096-1074.

18. Marcu D. 1997; From discourse sturctures to text summaries. Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable text Summarization:82-88.

19. Mann WC, Thompson SA. 1988; Rhetorical structure theory: towards a functional theory of text organization. Text 8(3):243-281.

20. Barzilay R, Elhadad M. 1997; Using lexical chanins for text summarization. Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97):10-17.

21. DeJong GG. An overview of the frump system. In: Lehnert W, Ringle M, editors. Strategies for Natural Language Processing. Hillsdale, NJ: Lawrence Erlbaum, 1982.

22. Jacobs PS, Rau LF. 1990; SCISOR: Extracting Information from On-Line News. Commun ACM 33(11):88-97.

23. Fiszman M., Rindflesch TC, Kilicoglu H. 2004; Summarization of an online medical encyclopedia. Medinfo:506-510.

24. Carbonell J, Goldstein J. 1998; The use of MMR, diversity-based reranking for reordering documents and producing summaries. Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval:335-336.

25. Masys DR, Welsh JB, Lynn Fink J, Gribskov M, Klacansky I, Corbeil J. 2001; Use of keyword hierarchies to interpret gene expression patterns. Bioinformatics 17:319-326.

26. Jenssen TK, Laegreid A, Komorowski J, Hovig E. 2001; A literature network of human genes for high-throughput analysis of gene expression. Nat Genet 28:21-28.

27. Chaussabel D, Sher A. 2002; Mining microarray expression data by literature profiling. Genome Biology 3(10):research0055.

28. Glenisson P, Antal P, Mathys J, Moreau Y, De Moor B. 2003; Evaluation of the vector space representation in text-based gene clustering. Pac Symp Biocomput:391-402.

29. Glenisson P, Coessens B, Van Vooren S, Mathys J, Moreau Y, De Moor B. 2004; TXTGate: profiling gene groups with text-based information. Genome Biol 5(6):r43.

30. Homayouni R, Heinrich K, Wei L, Berry MW. 2005; Gene clustering by Latent Semantic Indexing of MEDLINE abstracts. Bioinformatics 21(1):104-115.

31. Liu Y, Navathe SB. 2005; Text mining biomedical literature for discovering gene-to-gene relationships: a comparative study of algorithms. IEEE/ACM Trans Comput Biol Bioinform 2(1):62-74.

32. Raychaudhuri S, Schutze H, Altman RB. 2002; Using Text Analysis to Identify Functionally Coherent Gene Groups. Genome Res 12(10):1582-1590.

33. Raychaudhuri S, Chang JT, Imam F, Altman RB. 2003; The computational analysis of scientific literature to define and recognize gene expression clusters. Nucl Acids Res 31(15):4553-4560.

34. Kuffner R, Fundel K, Zimmer R. 2005; Expert knowledge without the expert: integrated analysis of gene expression and literature to derive active functional contexts. Bioinformatics 21(suppl_2):ii259-ii267.

35. Huang D, Pan W. 2006; Incorporating biological knowledge into distance-based clustering analysis of microarray gene expression data. Bioinformatics 22(10):1259-1268.

36. Liu Y, Ciliax BJ. 2004; Comparison of two schemes for automatic keyword extraction from MEDLINE for functional gene clustering. Proc IEEE Comput Syst Bioinform Conf:394-404.

37. Glenisson P, Mathys J, Moreau Y, De Moor B. 2003; Scoring and summarizing gene groups from text using the vector space model. Technical Report 03-97, ESAT-SISTA.

38. Kankar P, et al. 2002; MedMeSH Summarizer: Text Mining for Gene Clusters. In the Proceedings of the Second SIAM International Conference on Data Mining.

39. Tanabe L, Scherf U, Smith L, Lee J, Hunter L, weinstein J. 1999; MedMiner: an Internet text-mining tool for biomedical information, with application to gene expression profiling. Biotechniques 27(6):1210-1217.

40. Divoli A, Attwood TK. 2005; BioIE: extracting informative sentences from the biomedical literature. Bioinformatics 21(9):2138-2139.

41. Mitchell AL, Divoli A, Kim JH, Hilario M, Selimas I, Attwood TK. 2005; METIS: multiple extraction techniques for informative sentences. Bioinformatics 21(22):4196-4197.

42. Ling X, et al. 2005; Automatically generating gene summaries for biomedical literature. In: Proceedings of the Pacific Symposium on Bioinformatics.

43. Ding J, Berleant D, Nettleton D, Wurtele E. 2002; Mining MEDLINE: abstracts, sentences, or phrases? Pac Symp Biocomput:326-337.

44. Edmundson H. 1969; New methods in automatic extracting. Journal of the ACM 16(2):264-285.

45. Cohen A. 2005; Unsupervised gene/protein entity normalization using automatically extracted dictionaries. In: Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics, Proceedings of the BioLINK2005 Workshop; Detroit, MI: Association for Computational Linguistics; 2005:17-24.

46. Muller HM, Kenny EE, Sternberg PW. 2004; Textpresso: an ontology-based information retrieval and extraction system for biological literature. PLoS Biol 2(11(e309)):1984-1998.

47. Mani I. 2001; Summarization evaluation: an overview. In: Proceedings of the NTCIR Workshop 2 Meeting on valuation of Chinese and Japanese Text Retrieval and Text Summarization.

48. Labroche N., Monmarché N, Venturini G. 2002; A new clustering algorithm based on the chemical recognition system of ants. In: Proceedings of ECAI 2002:345-349.

49. Teufel S, Moens M. 1997; Sentence Extraction as a Classification Task. Workshop 'Intelligent and scalable Text summarization', ACL/EACL 1997, July 1997.

50. Chiang JH, Shin JW, Liu HH, Chin CL. 2006; GeneLibrarian: an effective gene-information summarization and visualization system. BMC Bioinformatics 7(1):392.

51. Hirschman L, Colosimo M, Morgan A, Yeh A. 2005; Overview of BioCreAtIvE task 1B: normalized gene lists. BMC Bioinformatics 6(Suppl 1):S11.

52. Wang JZ, Du Z, Payattakool R, Yu PS, Chen CF. 2007; A New Method to Measure the Semantic Similarity of GO Terms. Bioinformatics:btm087.

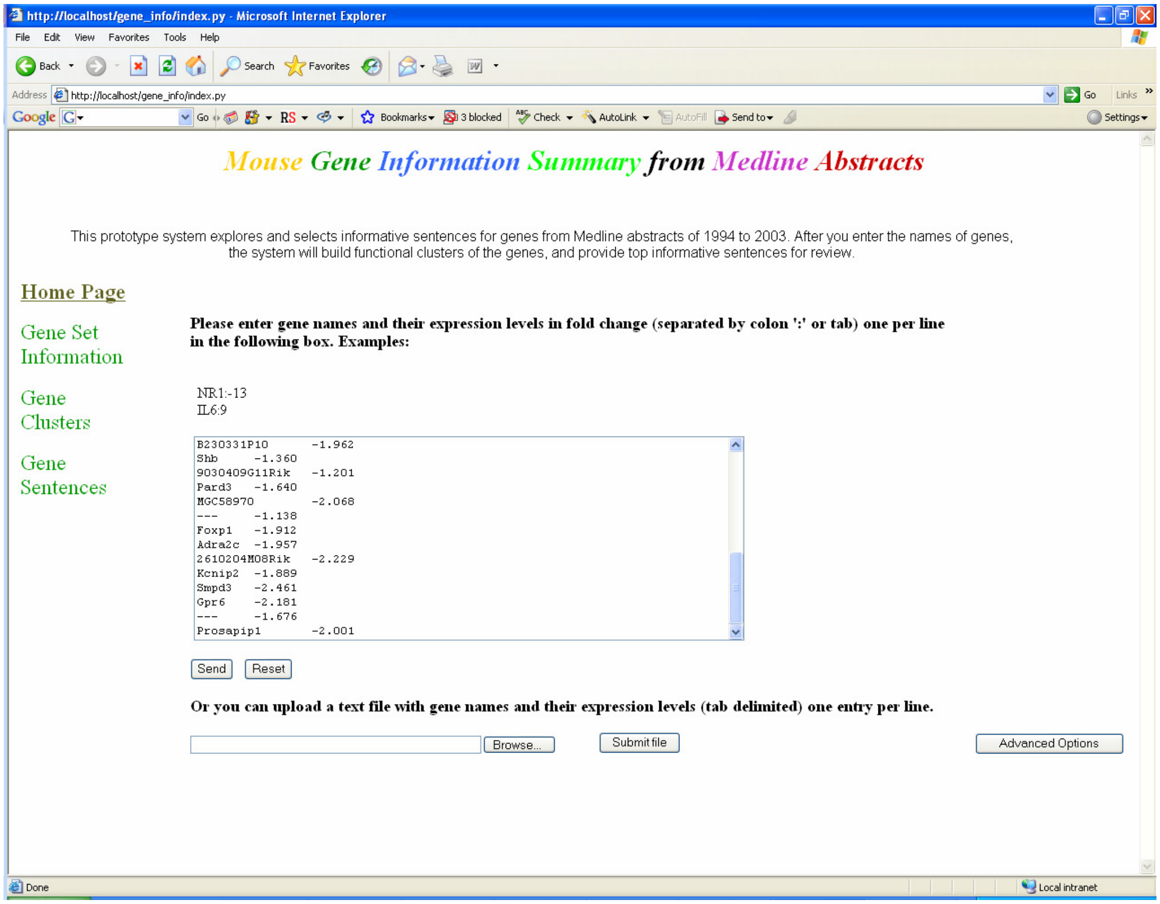
53. Andrade MA, Valencia A. 1998; Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families. Bioinformatics 14(7):600-607.

54. Chen H, Sharp B. 2004; Content-rich biological network constructed by mining PubMed abstracts. BMC Bioinformatics 5(1):147.

# Appendixes

1. Screen shots
2. Excluded English stop words
3. Cue words
4. GO terms associated with MGI_ID as provided by MGI
5. PubMed search queries for the ten genes
6. Sentence relevance judgment files
7. Major codes

## Appendix 1. Screen shots


http://localhost/gene_info/index.py - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://localhost/gene_info/index.py
Go
Links
Google
Bookmarks
3 blocked
Check
AutoLink
AutoFill
Send to
Settings
Done
Local intranet

*Mouse Gene Information Summary from Medline Abstracts*

This prototype system explores and selects informative sentences for genes from Medline abstracts of 1994 to 2003. After you enter the names of genes, the system will build functional clusters of the genes, and provide top informative sentences for review.

Home Page

Gene Set Information

Gene Clusters

Gene Sentences

**Please enter gene names and their expression levels in fold change (separated by colon ':' or tab) one per line in the following box. Examples:**

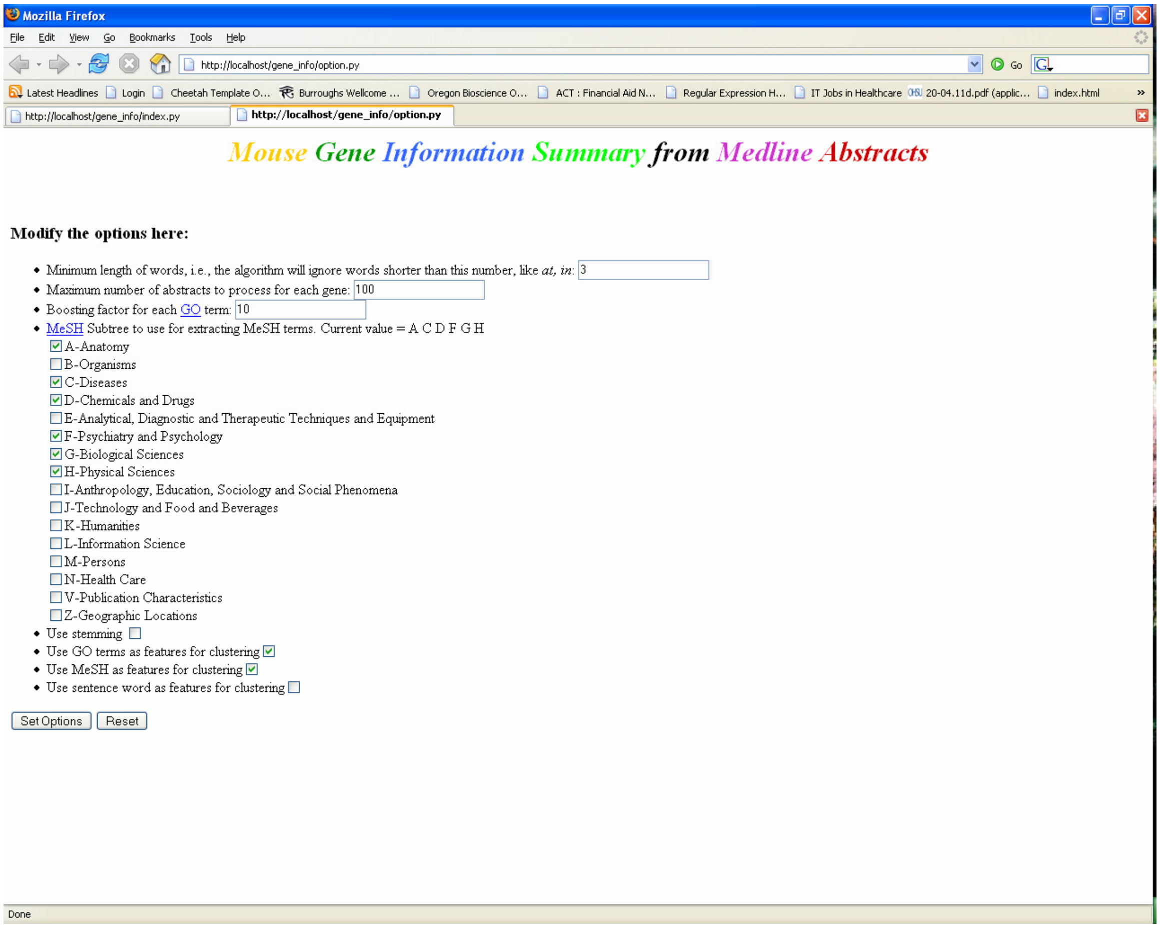
NR1:-13
IL6:9

```
B230331P10      -1.962
Shb     -1.360
9030409G11Rik   -1.201
Pard3   -1.640
MGC58970        -2.068
---     -1.138
Foxp1   -1.912
Adra2c  -1.957
2610204M08Rik   -2.229
Kcnip2  -1.889
Smpd3   -2.461
Gpr6    -2.181
---     -1.676
Prosapip1       -2.001
```

Send  Reset

**Or you can upload a text file with gene names and their expression levels (tab delimited) one entry per line.**

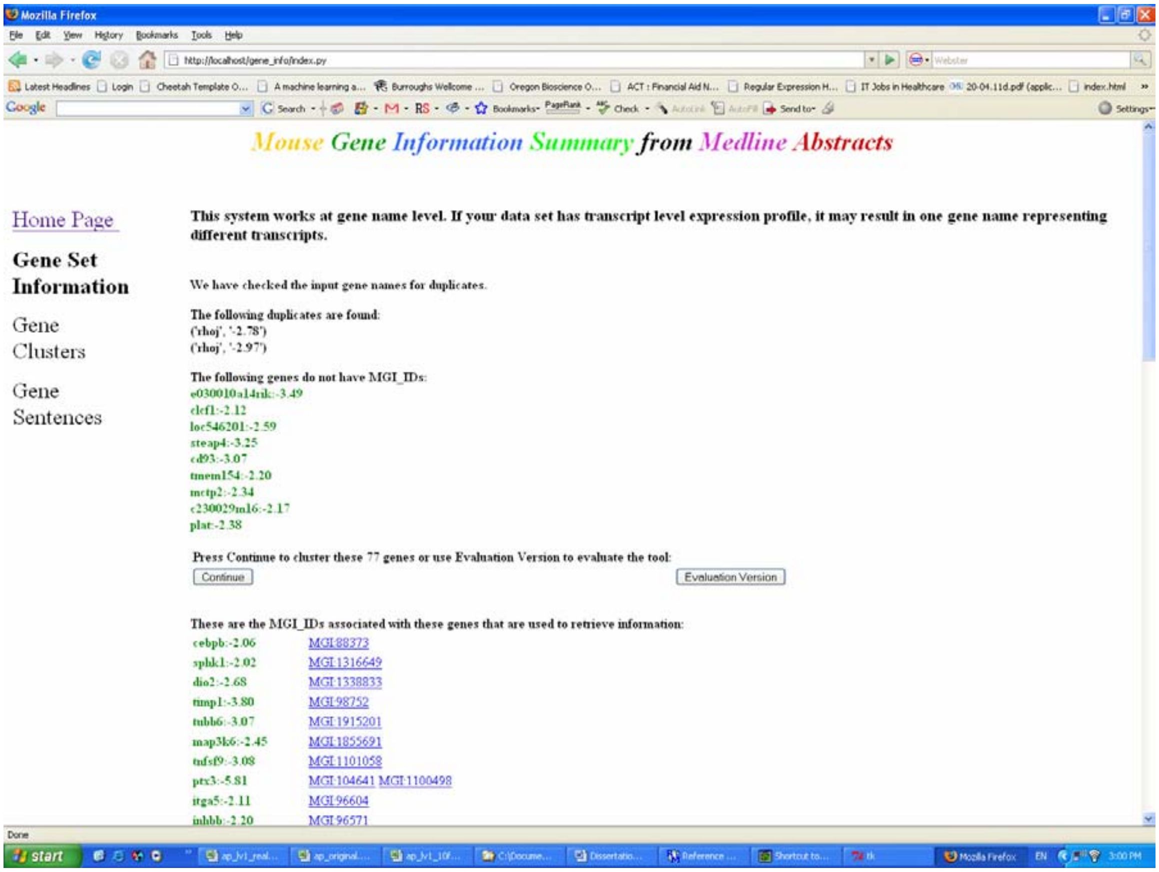
Browse...  Submit file  Advanced Options



*Mouse Gene Information Summary from Medline Abstracts*

**Modify the options here:**

- Minimum length of words, i.e., the algorithm will ignore words shorter than this number, like *at, in*: 3
- Maximum number of abstracts to process for each gene: 100
- Boosting factor for each GO term: 10
- MeSH Subtree to use for extracting MeSH terms. Current value = A C D F G H
  ☑ A-Anatomy
  □ B-Organisms
  ☑ C-Diseases
  ☑ D-Chemicals and Drugs
  □ E-Analytical, Diagnostic and Therapeutic Techniques and Equipment
  ☑ F-Psychiatry and Psychology
  ☑ G-Biological Sciences
  ☑ H-Physical Sciences
  □ I-Anthropology, Education, Sociology and Social Phenomena
  □ J-Technology and Food and Beverages
  □ K-Humanities
  □ L-Information Science
  □ M-Persons
  □ N-Health Care
  □ V-Publication Characteristics
  □ Z-Geographic Locations
- Use stemming □
- Use GO terms as features for clustering ☑
- Use MeSH as features for clustering ☑
- Use sentence word as features for clustering □


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/gene_info/index.py
Latest Headlines Login Cheetah Template O... A machine learning a... Burroughs Wellcome ... Oregon Bioscience O... ACT : Financial Aid N... Regular Expression H... IT Jobs in Healthcare 20-04.11d.pdf (applic... index.html
Google Search Bookmarks PageRank Check AutoLink AutoFill Send to Settings
Done
start

*Mouse Gene Information Summary* ***from*** *Medline Abstracts*

Home Page

**Gene Set Information**

Gene Clusters

Gene Sentences

**This system works at gene name level. If your data set has transcript level expression profile, it may result in one gene name representing different transcripts.**

**We have checked the input gene names for duplicates.**

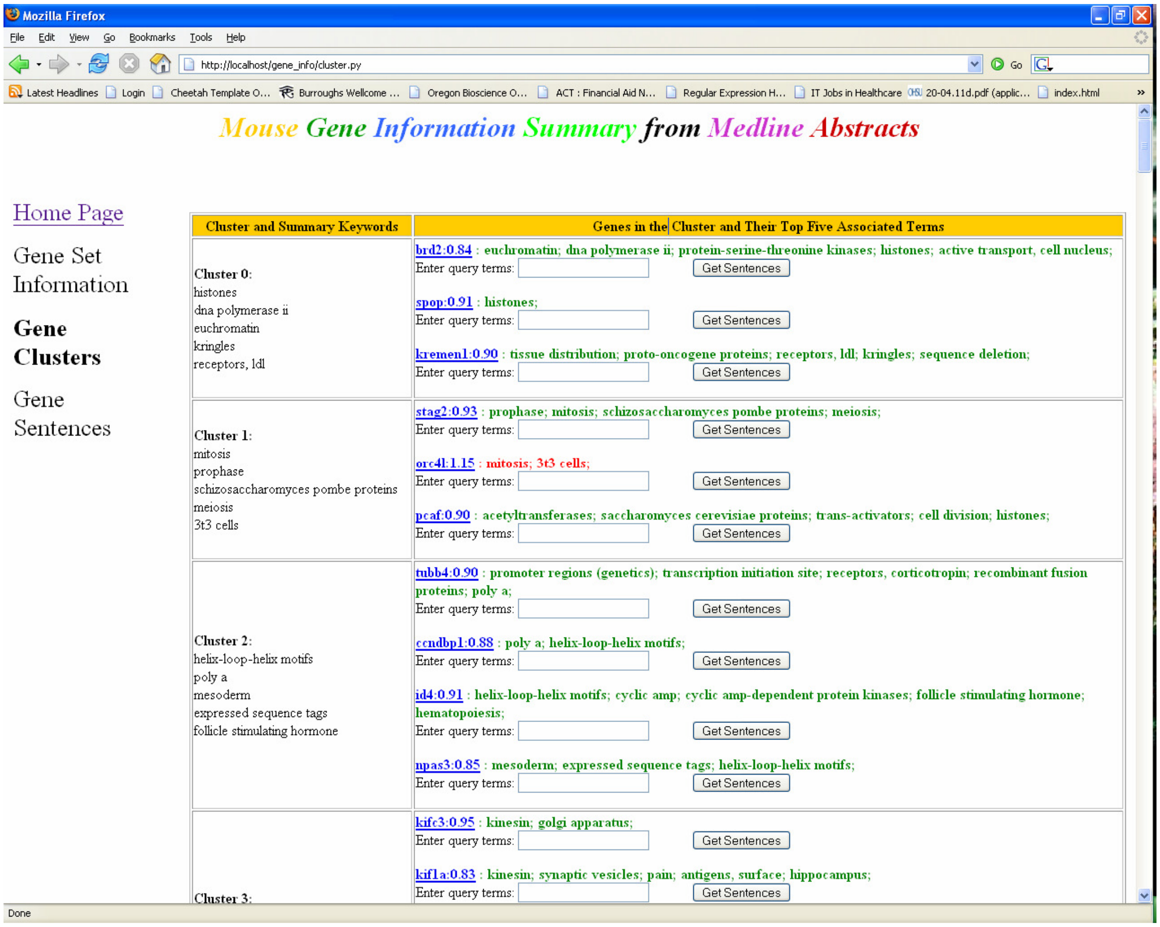
**The following duplicates are found:**
('rhoj', '-2.78')
('rhoj', '-2.97')

**The following genes do not have MGI_IDs:**
e030010a14rik:-3.49
clcf1:-2.12
loc546201:-2.59
steap4:-3.25
cd93:-3.07
tmem154:-2.20
mctp2:-2.34
c230029m16:-2.17
plat:-2.38

**Press Continue to cluster these 77 genes or use Evaluation Version to evaluate the tool:**

**These are the MGI_IDs associated with these genes that are used to retrieve information:**

| | |
|---|---|
| cebpb:-2.06 | MGI:88373 |
| sphk1:-2.02 | MGI:1316649 |
| dio2:-2.68 | MGI:1338833 |
| timp1:-3.80 | MGI:98752 |
| tubb6:-3.07 | MGI:1915201 |
| map3k6:-2.45 | MGI:1855691 |
| tnfsf9:-3.08 | MGI:1101058 |
| ptx3:-5.81 | MGI:104641 MGI:1100498 |
| itga5:-2.11 | MGI:96604 |
| inhbb:-2.20 | MGI:96571 |



*Mouse Gene Information Summary from Medline Abstracts*

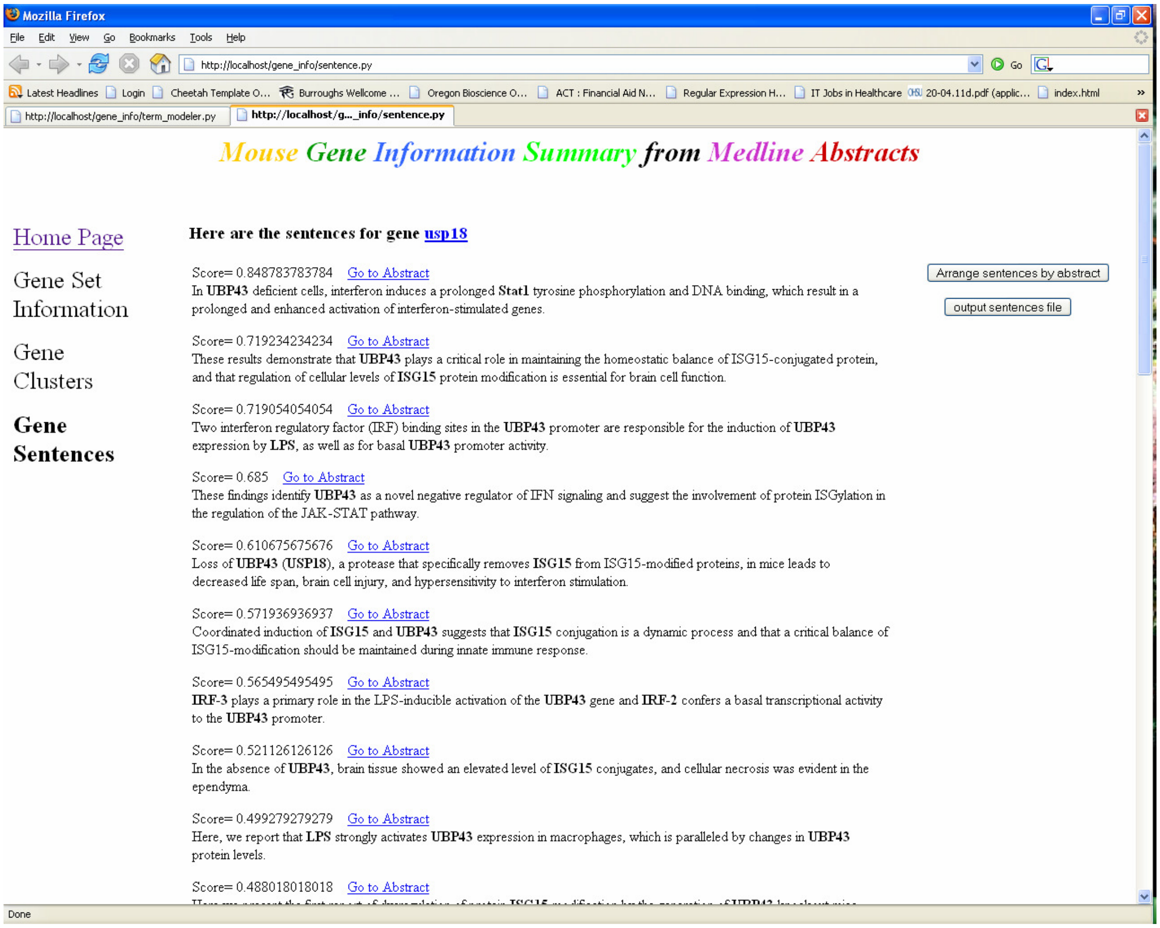
Home Page

Gene Set Information

**Gene Clusters**

Gene Sentences

| Cluster and Summary Keywords | Genes in the Cluster and Their Top Five Associated Terms |
|---|---|
| **Cluster 0**:<br>histones<br>dna polymerase ii<br>euchromatin<br>kringles<br>receptors, ldl | brd2:0.84 : euchromatin; dna polymerase ii; protein-serine-threonine kinases; histones; active transport, cell nucleus;<br>Enter query terms: Get Sentences<br>spop:0.91 : histones;<br>Enter query terms: Get Sentences<br>kremen1:0.90 : tissue distribution; proto-oncogene proteins; receptors, ldl; kringles; sequence deletion;<br>Enter query terms: Get Sentences |
| **Cluster 1**:<br>mitosis<br>prophase<br>schizosaccharomyces pombe proteins<br>meiosis<br>3t3 cells | stag2:0.93 : prophase; mitosis; schizosaccharomyces pombe proteins; meiosis;<br>Enter query terms: Get Sentences<br>orc4l:1.15 : mitosis; 3t3 cells;<br>Enter query terms: Get Sentences<br>pcaf:0.90 : acetyltransferases; saccharomyces cerevisiae proteins; trans-activators; cell division; histones;<br>Enter query terms: Get Sentences |
| **Cluster 2**:<br>helix-loop-helix motifs<br>poly a<br>mesoderm<br>expressed sequence tags<br>follicle stimulating hormone | tubb4:0.90 : promoter regions (genetics); transcription initiation site; receptors, corticotropin; recombinant fusion proteins; poly a;<br>Enter query terms: Get Sentences<br>ccndbp1:0.88 : poly a; helix-loop-helix motifs;<br>Enter query terms: Get Sentences<br>id4:0.91 : helix-loop-helix motifs; cyclic amp; cyclic amp-dependent protein kinases; follicle stimulating hormone; hematopoiesis;<br>Enter query terms: Get Sentences<br>npas3:0.85 : mesoderm; expressed sequence tags; helix-loop-helix motifs;<br>Enter query terms: Get Sentences |
| **Cluster 3**: | kifc3:0.95 : kinesin; golgi apparatus;<br>Enter query terms: Get Sentences<br>kif1a:0.83 : kinesin; synaptic vesicles; pain; antigens, surface; hippocampus;<br>Enter query terms: Get Sentences |



*Mouse Gene Information Summary from Medline Abstracts*

**Here are the sentences for gene usp18**

Score= 0.848783783784 Go to Abstract
In **UBP43** deficient cells, interferon induces a prolonged **Stat1** tyrosine phosphorylation and DNA binding, which result in a prolonged and enhanced activation of interferon-stimulated genes.

Score= 0.719234234234 Go to Abstract
These results demonstrate that **UBP43** plays a critical role in maintaining the homeostatic balance of ISG15-conjugated protein, and that regulation of cellular levels of **ISG15** protein modification is essential for brain cell function.

Score= 0.719054054054 Go to Abstract
Two interferon regulatory factor (IRF) binding sites in the **UBP43** promoter are responsible for the induction of **UBP43** expression by **LPS**, as well as for basal **UBP43** promoter activity.

Score= 0.685 Go to Abstract
These findings identify **UBP43** as a novel negative regulator of IFN signaling and suggest the involvement of protein ISGylation in the regulation of the JAK-STAT pathway.

Score= 0.610675675676 Go to Abstract
Loss of **UBP43** (**USP18**), a protease that specifically removes **ISG15** from ISG15-modified proteins, in mice leads to decreased life span, brain cell injury, and hypersensitivity to interferon stimulation.

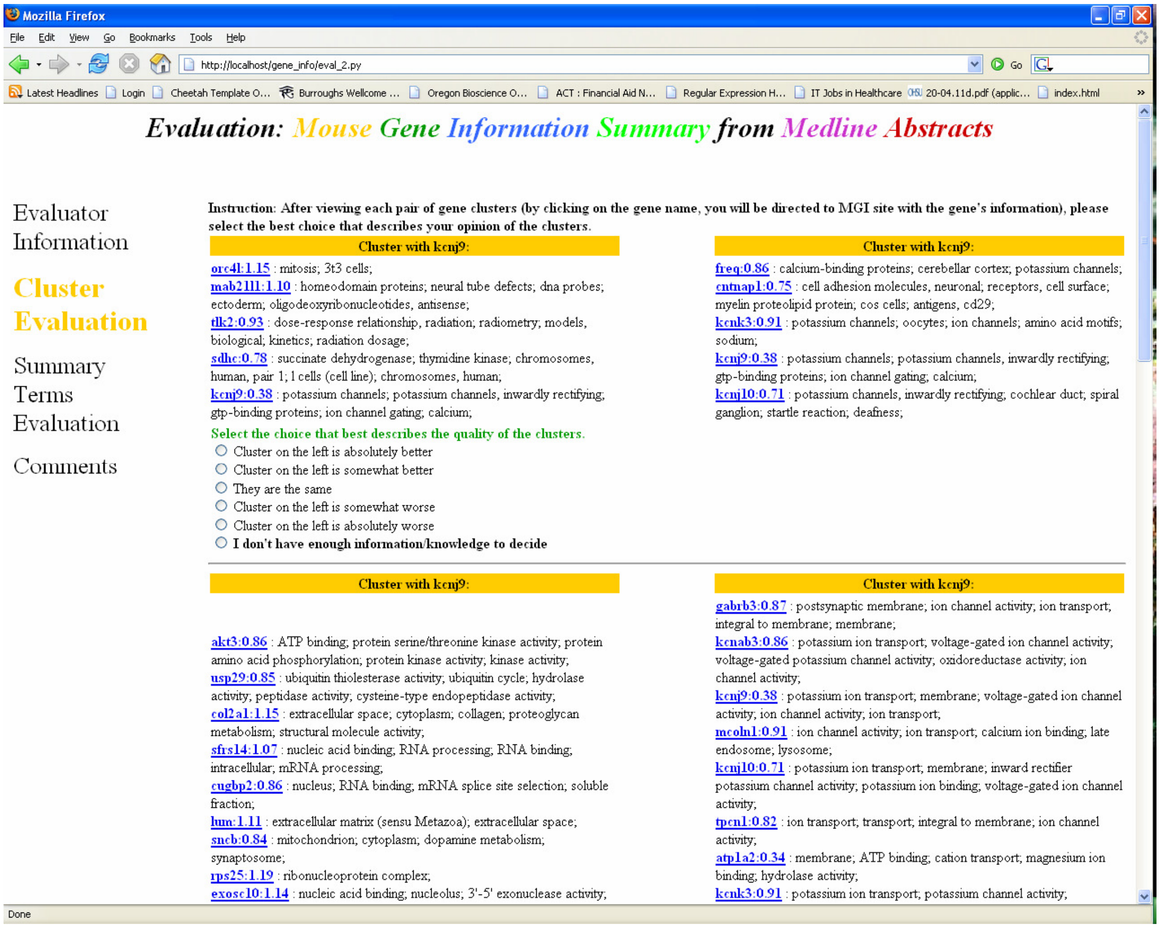
Score= 0.571936936937 Go to Abstract
Coordinated induction of **ISG15** and **UBP43** suggests that **ISG15** conjugation is a dynamic process and that a critical balance of ISG15-modification should be maintained during innate immune response.

Score= 0.565495495495 Go to Abstract
**IRF-3** plays a primary role in the LPS-inducible activation of the **UBP43** gene and **IRF-2** confers a basal transcriptional activity to the **UBP43** promoter.

Score= 0.521126126126 Go to Abstract
In the absence of **UBP43**, brain tissue showed an elevated level of **ISG15** conjugates, and cellular necrosis was evident in the ependyma.

Score= 0.499279279279 Go to Abstract
Here, we report that **LPS** strongly activates **UBP43** expression in macrophages, which is paralleled by changes in **UBP43** protein levels.

Score= 0.488018018018 Go to Abstract



*Evaluation: Mouse Gene Information Summary from Medline Abstracts*

**Instruction: After viewing each pair of gene clusters (by clicking on the gene name, you will be directed to MGI site with the gene's information), please select the best choice that describes your opinion of the clusters.**

**Cluster with kcnj9:**

**orc4l:1.15** : mitosis; 3t3 cells;
**mab21l1:1.10** : homeodomain proteins; neural tube defects; dna probes; ectoderm; oligodeoxyribonucleotides, antisense;
**tlk2:0.93** : dose-response relationship, radiation; radiometry; models, biological; kinetics; radiation dosage;
**sdhc:0.78** : succinate dehydrogenase; thymidine kinase; chromosomes, human, pair 1; l cells (cell line); chromosomes, human;
**kcnj9:0.38** : potassium channels; potassium channels, inwardly rectifying; gtp-binding proteins; ion channel gating; calcium;

**Cluster with kcnj9:**

**freq:0.86** : calcium-binding proteins; cerebellar cortex; potassium channels;
**cntnap1:0.75** : cell adhesion molecules, neuronal; receptors, cell surface; myelin proteolipid protein; cos cells; antigens, cd29;
**kcnk3:0.91** : potassium channels; oocytes; ion channels; amino acid motifs; sodium;
**kcnj9:0.38** : potassium channels; potassium channels, inwardly rectifying; gtp-binding proteins; ion channel gating; calcium;
**kcnj10:0.71** : potassium channels, inwardly rectifying; cochlear duct; spiral ganglion; startle reaction; deafness;

**Select the choice that best describes the quality of the clusters.**

○ Cluster on the left is absolutely better
○ Cluster on the left is somewhat better
○ They are the same
○ Cluster on the left is somewhat worse
○ Cluster on the left is absolutely worse
○ **I don't have enough information/knowledge to decide**

**Cluster with kcnj9:**

**akt3:0.86** : ATP binding; protein serine/threonine kinase activity; protein amino acid phosphorylation; protein kinase activity; kinase activity;
**usp29:0.85** : ubiquitin thiolesterase activity; ubiquitin cycle; hydrolase activity; peptidase activity; cysteine-type endopeptidase activity;
**col2a1:1.15** : extracellular space; cytoplasm; collagen; proteoglycan metabolism; structural molecule activity;
**sfrs14:1.07** : nucleic acid binding; RNA processing; RNA binding; intracellular; mRNA processing;
**cugbp2:0.86** : nucleus; RNA binding; mRNA splice site selection; soluble fraction;
**lum:1.11** : extracellular matrix (sensu Metazoa); extracellular space;
**sncb:0.84** : mitochondrion; cytoplasm; dopamine metabolism; synaptosome;
**rps25:1.19** : ribonucleoprotein complex;
**exosc10:1.14** : nucleic acid binding; nucleolus; 3'-5' exonuclease activity;

**Cluster with kcnj9:**

**gabrb3:0.87** : postsynaptic membrane; ion channel activity; ion transport; integral to membrane; membrane;
**kcnab3:0.86** : potassium ion transport; voltage-gated ion channel activity; voltage-gated potassium channel activity; oxidoreductase activity; ion channel activity;
**kcnj9:0.38** : potassium ion transport; membrane; voltage-gated ion channel activity; ion channel activity; ion transport;
**mcoln1:0.91** : ion channel activity; ion transport; calcium ion binding; late endosome; lysosome;
**kcnj10:0.71** : potassium ion transport; membrane; inward rectifier potassium channel activity; potassium ion binding; voltage-gated ion channel activity;
**tpcn1:0.82** : ion transport; transport; integral to membrane; ion channel activity;
**atp1a2:0.34** : membrane; ATP binding; cation transport; magnesium ion binding; hydrolase activity;
**kcnk3:0.91** : potassium ion transport; potassium channel activity;

Appendix 2. Excluded English stop words can be accessed at

http://ir.ohsu.edu/~jianji/gene_info/static/english.stop

Appendix 3. Cue words can be accessed at

http://ir.ohsu.edu/~jianji/gene_info/static/key_terms.txt

.

Appendix 4. Table of GO terms that are associated with MGI_ID form MGI.

Can be accessed at http://ir.ohsu.edu/~jianji/gene_info/static/MGI_GO

Appendix 5. Queries for PubMed searches for the ten genes.

**Cyp2j5**:
http://www.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=Pubmed&retmax=1&usehistory=y&mindate=1994&maxdate=2003&sort=pub+date&term=cyp2j5+OR+cyp2j-5

**frk:**
http://www.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=Pubmed&retmax=1&usehistory=y&mindate=1994&maxdate=2003&sort=pub+date&term=frk+OR+c85044+OR+bsk/iyk+OR+bsk+OR+rak+OR+c-85044+OR+gtk

**kcnj9:**
http://www.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=Pubmed&retmax=1&usehistory=y&mindate=1994&maxdate=2003&sort=pub+date&term=girk-3+OR+girk3+OR+kcnj9+OR+kcnj-9+OR+mbgirk3+OR+mbgirk-3+OR+kir3.3

**pglyrp1:**
http://www.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=Pubmed&retmax=1&usehistory=y&mindate=1994&maxdate=2003&sort=pub+date&term=tag7+OR+tnfsf-3l+OR+pgrp+OR+pgrps+OR+pglyrp-1+OR+tasg7+OR+tasg-7+OR+pgrp-s+OR+pglyrp1+OR+pglyrp+OR+tnfsf3l+OR+tag-7

**adamts1:**
http://www.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=Pubmed&retmax=1&usehistory=y&mindate=1994&maxdate=2003&sort=pub+date&term=adamts1+OR+c3c5+OR+meth1+OR+adamts+OR+adamts5+OR+adamts11+OR+asmp2+OR+implantin+OR+ai481094

**clca1:**
http://www.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=Pubmed&retmax=1&usehistory=y&mindate=1994&maxdate=2003&term=clca1+OR+cacc

**cxcl12:**
http://www.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=Pubmed&retmax=1&usehistory=y&mindate=1994&maxdate=2003&sort=pub+date&term=sdf-1alpha+OR+sdf1-b+OR+sdf1-a+OR+sdf-1b+OR+sdf-1a+OR+sdf-1-b+OR+sdf-1-a+OR+pbsf/sdf1+OR+pbsf+OR+ai-174028+OR+sdf1b+OR+sdf1a+OR+tpar1+OR+sdf1+OR+sdf1beta+OR+ai174028+OR+scyb12+OR+sdf-1-alpha+OR+cxcl12+OR+sdf1-alpha+OR+sdf1-beta+OR+cxcl-12+OR+tlsf+OR+sdf-1beta+OR+pbsf/sdf-1+OR+tlsf-b+OR+tlsf-a+OR+sdf-1+OR+tpar-1+OR+tlsfa+OR+tlsfb+OR+scyb-12+OR+sdf1alpha+OR+sdf-1-beta

**irak3:**
http://www.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=Pubmed&retmax=1&usehistory=y&mindate=1994&maxdate=2003&sort=pub+date&term=ai-563835+OR+irak-m+OR+irak-3+OR+ai563835+OR+4833428c18rik+OR+irak3+OR+irakm

**ptx3:**
http://www.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=Pubmed&retmax=1&usehistory=y&mindate=1994&maxdate=2003&sort=pub+date&term=pitx-3+OR+pitx3+OR+ptx-3+OR+ptx3

**usp18:**
http://www.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=Pubmed&retmax=1&usehistory=y&mindate=1994&maxdate=2003&term=usp18+OR+1110058h21rik+OR+aw047653+OR+ubp43+OR+ubp15

Appendix 6 Sentence relevance judgment files can be access at:

http://ir.ohsu.edu/~jianji/gene_info/eval/sentence

Appendix 7. Major codes

########## Index.py
########## entry point: get input from user by uploading file

```
import os,tempfile
from mainhtml import *
from localUtil import *
from option_class import *
from checkRec import *

if not QUERY:
  option=options(1,['A','C','D','F','G','H'],3,100,0,0,1,5)
  Session().option=option
  print mainhtml

elif not _filename=='':

    f = _filename.file # file-like object
    dest_name = tempfile.mktemp(dir='.')
    out = open(dest_name,'wb')
    # copy file
    import shutil
    shutil.copyfileobj(f,out)
    out.close()
    infile=open(dest_name,'r')

    duplicate=[]
    genes={}
    for line in infile.readlines():
      if line.strip():
        geneS,fold=line.strip().lower().split('\t')
        gene=geneS.strip()
        if genes.get(gene,0):
          duplicate.append((gene,genes[gene]))
          duplicate.append((gene,fold))
        else:
          genes[gene]=fold

    infile.close()
    os.remove(dest_name)

    for g in genes:
      if float(genes[g])<0:
        genes[g]=(genes[g],'green')
      else:
        genes[g]=(genes[g],'red')

    gene_id, no_record=check_no_record(genes.keys())

    Session().gene_id=gene_id
    Session().genes=genes
    data = {
    'duplicate': duplicate,
    'gene_id': gene_id,
    'no_record':no_record,
    'count':len(gene_id),
    'genes':genes}

    merge('second.html', data)

else:
    print errPage2
```

```
###### Index2.py
###### entry point, get input from user by manual entering of genes

import sys
from mainhtml import *
from localUtil import *
from checkRec import *

def error_exit():
    print errPage1

def parse_input():
    genes={}
    duplicate=[]
    firstline=1
    for item in list(_names[0].strip().split('\r\n')):
      if firstline:
        if ':' in item:
          sep=':'
          firstline=0
        elif '\t' in item:
          sep='\t'
          firstline=0
        else:
          error_exit()
          return

      geneS,fold=item.strip().lower().split(sep)
      gene=geneS.strip()
      if genes.get(gene,0):
        duplicate.append((gene,genes[gene]))
        duplicate.append((gene,fold))
      else:
        genes[gene]=fold
    for g in genes.keys():
        if float(genes[g])<0:
          genes[g]=(genes[g],'green')
        else:
          genes[g]=(genes[g],'red')

    gene_id, no_record=check_no_record(genes.keys())

    Session().gene_id=gene_id
    Session().genes=genes

    data = {
    'duplicate': duplicate,
    'gene_id': gene_id,
    'no_record':no_record,
    'count':len(gene_id),
    'genes':genes}

    merge('second.html', data)
if not _names[0]:
    error_exit()

else:
    parse_input()
```

```
######  term_modeler.py
###### build gene model using GO, mesh and text word

import os, tempfile
from localUtil import *
from cluto import *
from GOTerm import *
from mesh_text import *

gene_id=Session().gene_id
maxAbs=Session().option.maxAbs
meshSub=Session().option.meshSub
stem=Session().option.stem
minWordLen=Session().option.minWordLen



delta=0.035
inc=0.01
gene_term={}   # dict with key gene name, value is dict of go term counts
mesh_term={}
no_record=[]

if Session().option.GO:
  gene_term.update(get_go_term(gene_id,Session().option.goFactor))

mesh_term, gene_pmid=get_mesh(gene_id,meshSub,maxAbs,minWordLen,stem)
# need to separate the mesh subcategory selection from inside get_mesh in
localUtil!!!!!

Session().gene_pmid=gene_pmid

# merging go and mesh terms

if Session().option.mesh:
    gene_term.update(mesh_term)

# if using word features
if Session().option.wordFeature:
    word_term=get_geneWord(gene_pmid,minWordLen,stem)
    gene_term.update(word_term)

# find genes with no record ({} gene_terms)
for gene in gene_id.keys():
    if not gene_term.has_key(gene):
      no_record.append(gene)
    elif not gene_term[gene]:
      no_record.append(gene)
      del gene_term[gene]

# output cluto datafiles

mat, clabel, genename, top_terms=outputMatrixFile(gene_term)
Session().top_terms=top_terms

if len(gene_term)<6:
    os.chdir('cluto')
    os.remove(mat)
    os.remove(clabel)
    os.remove(genename)
    os.chdir('..')
    Session().option.GO=0
    data={
      'top_terms':top_terms,
      'id':Session().gene_id,
      'genes':Session().genes,
      'no_record':no_record}
    merge('noclustering.html',data)
else:
# clustering
  os.chdir('cluto')
  out=tempfile.mktemp(dir='.')
  cluster=tempfile.mktemp(dir='.')

  cluto(delta, inc,len(gene_term),cluster,mat, out,clabel)

  os.remove(mat)
  os.remove(clabel)
  os.chdir('..')
# pass data to process cluto output

  data = {
    'out': out,
    'genes':Session().genes,
    'no_record':no_record,
    'id':Session().gene_id,
    'cluster':cluster,
    'genename':genename}

  merge('third.html', data)
```

```
####### cluster.py
####### process cluto output

from proc_cluster import *
from localUtil import *

os.chdir('cluto')
cluster_gene, desc_word=proc_cluster(_genename,_cluster,_out)
os.remove(_genename)
os.remove(_cluster)
os.remove(_out)
os.chdir('..')
# merge with html file
sort_list=[int(c) for c in cluster_gene.keys()]
sort_list.sort()
sort_list=[str(c) for c in sort_list]

# use " to subsitute ' in desc_word
sub_desc={}
for key in desc_word.keys():
  sub_desc[key]=[w.replace("'",'@') for w in desc_word[key]]


data = {
    'gene_pmid':Session().gene_pmid.keys(),
    'genes':Session().genes,
    'sort_list':sort_list,
    'cluster_gene': cluster_gene,
    'sub_desc': sub_desc,
    'id':Session().gene_id,
    'topTerms':Session().top_terms,
    'desc_word':desc_word}

merge('fourth.html', data)
```

```
##### proc_cluster.py
##### functions for cluster processing
import os

def proc_cluster(genename,cluster,out):
    g_file=open(genename,'r')
    c_file=open(cluster,'r')
    o_file=open(out,'r')

    g_list=g_file.readlines()
   # print g_list,'<br>'
    c_list=c_file.readlines()
   # print c_list,'<br>'
    cluster_gene={}
    for i in range(0,len(g_list)):
      a=cluster_gene.get(c_list[i].strip(),[])
      a.append(g_list[i].strip())
      cluster_gene[c_list[i].strip()]=a
#print cluster_gene

    n=0
    desc_word={}
    desc_word['-1']=['Unable to cluster these genes']
    for line in o_file.readlines():
      if line[:7]=='Cluster':
        cluster=line[7:11].strip()
    #    print cluster
        n=1

        continue
      if n:
        words=line.strip().split(':')[1].split('%,')
        words=[w[:-5].strip() for w in words]
        desc_word[cluster]=words
        n=0
        continue

    g_file.close()
    c_file.close()
    o_file.close()

    return cluster_gene, desc_word
```

```
##### cluto.py
##### CLUTO wrapper

import os,tempfile
def outputMatrixFile(gene_words):
    os.chdir('cluto')
    matrix = tempfile.mktemp(dir='.')
    cl=tempfile.mktemp(dir='.')
    gname=tempfile.mktemp(dir='.')

    mfile = open(matrix,'w')
    cfile=open(cl,'w')
    gfile=open(gname,'w')

    clabel=[]
    count=0
    mat=[]
    top_terms={}
    for g in gene_words.keys():
        sort_terms=sorted(gene_words[g].items(), lambda x, y: cmp(y[1], x[1]))
        top_terms[g]=[t[0] for t in sort_terms[:5]]
        clabel.extend(gene_words[g].keys())
        clabel=list(set(clabel))
    mat.append(str(len(gene_words))+'\t'+str(len(clabel))+'\t')
    for g in gene_words.keys():
     #   print g
        gfile.write(str(g)+'\n')
        line=''
        for i in range(0,len(clabel)):
            if gene_words[g].get(clabel[i],0):
                count+=1
                line=line+str(i+1)+'\t'+str(gene_words[g][clabel[i]])+'\t'
        line=line.strip()+'\n'
        mat.append(line)
#    print len(mat)
    mat[0]=mat[0]+str(count)+'\n'
    for c in clabel:
        cfile.write(c+'\n')
    for m in mat:
        mfile.write(m)
    cfile.close()
    mfile.close()
    gfile.close()
    os.chdir('..')
    return matrix, cl, gname, top_terms

def find_best_n(delta, n,cluster,mat,out):
    i1=1
    for k in range(n/10+1,n/2,1):
        os.system('"./vcluster" -rowmodel=maxtf -clmethod=direct -clustfile=%s
%s %i > %s'%(cluster,mat,k,out))
        for line in open(out,'r').readlines():
            if '[' in line:
                r=line.split('[')[1]
                i2=float(r[3:-2])

                if (i2-i1)/i1 < delta:
                    return k
                else:
                    i1=i2
                    break

def cluto(delta, inc, maxclu,cluster,mat,out, clabel):
#test for cluster size
  n_clu=0

  while not n_clu:
    n_clu=find_best_n(delta,maxclu,cluster,mat,out)
    delta+=inc

#truc clustering
  os.system('"./vcluster" -colmodel=idf-rowmodel=maxtf -clabelfile=%s -
showfeatures -nfeatures=10 -clmethod=direct -clustfile=%s %s %i >
%s'%(clabel,cluster,mat,n_clu,out))

def cluto_e(n,cluster,mat,out,clabel):
       os.system('"./vcluster" -colmodel=idf -rowmodel=maxtf -clabelfile=%s -
showfeatures -clmethod=direct -clustfile=%s %s %i >
%s'%(clabel,cluster,mat,n,out))
```

```
###### sentence.py
###### score sentences by ranking scheme

from __future__ import division
from sent_sup import *
from localUtil import *


desc_word=eval(_desc_word)
desc_word=[w.replace('@',"'") for w in desc_word]
porter=PorterStemmer()
minWordLen=Session().option.minWordLen
meshSub=Session().option.meshSub
stem=1
# S=a CluSim + b QuFreq + c NGene + d CTword + e TPword +f L + g Recent
lengthFactor=20  # sentence more than 20 words will get L=1, else: scale to 0-1
a=0.2
b=0.1
c=0.1
d=0.0
e=0.4
f=0.0
g=0.2

if not Session().gene_pmid.has_key(_gene):
  data = {
    'id':Session().gene_id[_gene],
    'gene': _gene}

  merge('no_sent.html', data)

else:
  # calculate score
  # first get sentence words and query terms
  s_id= Session().gene_pmid[_gene]

# calculate recency
  recent={}
  inc=1/(len(s_id)-1)
  for i in range(0,len(s_id)):
    recent[s_id[i]]=1-i*inc

  sent_word, geneList, sentence=get_word(s_id,minWordLen,stem)
  # process nGene
  Gene=[]
  gene_id_list=[Gene.append(i) for i in Session().gene_id.values()]
  gene_id_dic=dict(zip(gene_id_list,[1]*len(gene_id_list)))
  nGene={}
  for s in geneList:
    if intersect(geneList[s],gene_id_dic)>1:
      nGene[s]=1
    elif len(geneList[s])>1:
      nGene[s]=.5
    else:
      nGene[s]=0
  #  print sentence[s],nGene[s],'\n'
  # process query terms
  if _query:
    query=_query.strip().split()
    query=[porter.stemWord(w) for w in query]

  CT_dict=CTword()
  TP_dict=TPword()
  score={}
  clusim={}
  clusimMax=1
  length={}
  ctword={}
  ctwordMax=1
  tpword={}
  tpwordMax=1
  for s in s_id:
    # clusim
    mesh=mesh_proc(s[0],meshSub,minWordLen,stem)
    m_clusim=intersect(desc_word,mesh)
    # need to add GO term sim too!!!!!!!!!!!!!!!!
    d_clusim=0
    if Session().option.GO:
      desc_ls=parse_desc(desc_word,minWordLen)
      d_clusim=intersect(desc_ls,sent_word[s])
    clusim[s]=m_clusim+d_clusim
    if clusimMax < clusim[s]:
      clusimMax=clusim[s]
    # quFreq
    qufreq=0
    if _query:
      qufreq=intersect(query,sent_word[s])

    #length
    sum=0
    for w in sent_word[s]:
      sum+=sent_word[s][w]
    if sum/lengthFactor<1:
      length[s]=sum/20
    else:
      length[s]=1
    #CTwords

    key=0
    sent_stem=sent_word[s].keys()
    for w in sent_stem:
        if CT_dict.has_key(w):
            key+=1
    #        print w
    ctword[s]=key
    if ctwordMax < key:
      ctwordMax=key
    #TPwords

    key=0
    for w in sent_stem:
      if TP_dict.has_key(w):
    #      print w
          key+=1
    tpword[s]=key
    if tpwordMax < key:
      tpwordMax=key
  for s in s_id:
    clusim[s]=clusim[s]/clusimMax
    ctword[s]=ctword[s]/ctwordMax
    tpword[s]=tpword[s]/tpwordMax

  #**************** output sentence list************

  for s in s_id:
    score[s]=a*clusim[s] + b*qufreq + c*nGene[s] + d*ctword[s] + e*tpword[s]
+f*length[s] + g*recent[s]

  # sorting by score
  sort_score=sorted(score.items(), lambda x, y: cmp(y[1], x[1]))

  Session().sort_score=sort_score
  Session().sentence=sentence



  data = {
    'gene':_gene,
    'id':Session().gene_id[_gene],
    'sort_score':sort_score,
    'sentence':sentence}

  merge('fifth.html', data)
```

```
#### sortByAbstract.py
#### to arrange the sentence first by abstract then by score

from localUtil import *
sortAbs=[]
sort_score=Session().sort_score
while sort_score:
  abs=[]
  top=sort_score.pop(0)
  abs.append(top)
  for s in sort_score:
    if s[0][0]==top[0][0]:
      below=sort_score.pop(sort_score.index(s))
      abs.append(below)
  sortAbs.append(abs)

data = {
    'gene':_gene,
    'sortAbs':sortAbs,
    'id':Session().gene_id[_gene],
    'sentence':Session().sentence}

merge('sixth.html', data)
```

```
###### sent_sup.py
###### sentence process fuctions

import os,re
from text_proc import *
conn=DB_conn('static/mice_gene')

def intersect(list1,dic2):
  count=0
  for l in list1:
    if dic2.has_key(l):
      count+=1
  return count

def parse_desc(desc_word,minWordLen):
  porter=PorterStemmer()
  stop_word=open('static/english.stop','r').readlines()
  stop_word=[s.strip() for s in stop_word]
  stopword=dict(zip(stop_word,[1]*len(stop_word)))
  desc_dic={}
  for t in desc_word:
    terms=t.split()
    for word in terms:
      if stopword.get(word,0) or len(word)<minWordLen or word.isdigit() or not
word.isalnum():
        continue
      else:
        desc_dic[word]=1
    terms=[porter.stemWord(w) for w in desc_dic.keys()]
  return terms

def mesh_proc(p,mesh_subtree,minWordLen,stem):
    sql="""select mesh from pmid_dp where pmid='%s';"""
    meshwords=conn.query(sql%(p))[0][0]
    if meshwords:
      proc=word_proc(minWordLen,stem)
      return proc.mesh({},meshwords,mesh_subtree)
    else:
      return mesh_dic

def get_word(sent_id,min_word_len,stem):
    matchstr=re.compile(r'<GENE_PROTEIN id="MGI:(\d)+">')
    matchstr2=re.compile(r'</GENE_PROTEIN>')
    processor=word_proc(min_word_len,stem)
    sql="""select sentence from pmid_sentence where pmid='%s' and
sent_pos='%i';"""
    sql2="""select MGI_gene from gene_pmid_pos where pmid='%s' and
sent_pos='%s';"""

    sent_word={}
    ngene={}
    sentence={}
    for s in sent_id:
      sent=conn.query(sql%(s[0],s[1]))[0][0]
      sentForDisp=matchstr.sub(r'<B>',sent)
      sentForDisp=matchstr2.sub(r'</B>',sentForDisp)
      sentence[s]=sentForDisp
      genes=conn.query(sql2%(s[0],s[1]))
      genes=[g[0] for g in genes]
      ngene[s]=genes
      sent_word[s]=processor.sentence_dist(sent)
  #    rows=conn.query(sql2%(s[0],s[1]))
  #    ngene[s]=rows[0][0]
    return sent_word, ngene, sentence


def CTword():
    key_word=open('static/key_terms.txt','r').readlines()
    p=PorterStemmer()
    key_word=[p.stemWord(s.strip()) for s in key_word]
    return dict(zip(key_word,[1]*len(key_word)))

def TPword():

files=['static/p_associte.txt','static/p_effect.txt','static/p_involvement.txt'
,'static/p_pathway.txt','static/p_purpose.txt','static/p_regulation.txt']
    tpword={}
    for f in files:
      key_word=open(f,'r').readlines()
      key_word=[s.strip() for s in key_word]
      tpword.update(dict(zip(key_word,[1]*len(key_word))))
    return tpword
```

```
###### text_proc.py
###### general text processing database connection functions.

from PorterStemmer import *

class synonym:
    def __init__(self,f_name,syn={}):
        infile=open(f_name,'r')
        self.syn_dict=syn
        for line in infile.readlines():
            lst=line.strip().split('\t')
            self.syn_dict[lst[0]]=lst[1]
            infile.close()
    def get_value(self,key):
        return self.syn_dict.get(key,[])

class DB_conn:
    def __init__(self,db_name):
        from pysqlite2 import dbapi2 as sqlite
        con=sqlite.connect(db_name)
        self.cursor=con.cursor()
    def query(self, sql):
        self.cursor.execute(sql)
        return self.cursor.fetchall()

class input_gene_list:
    def __init__(self):
        self.gene_list=[]

    def get_list(self, name_list):
        for item in name_list:
            item=item.strip().lower()
            if item and item not in self.gene_list:
                self.gene_list.append(item)
        return self.gene_list

class word_proc:
    def __init__(self,m,stem):
        self.min_word=m
        self.mesh_tree={}
 #       print 'loading stop words======='
        stop_word=open('static/english.stop','r').readlines()
        stop_word=[s.strip() for s in stop_word]
        self.stopword=dict(zip(stop_word,[1]*len(stop_word)))
  #      print 'loading mesh tree====='
        lines=open('static/mtrees2005.bin','r').readlines()
        for line in lines:
            words=line.split(';')
            self.mesh_tree[words[0]]=words[1].strip()
        self.stem=stem
    def words(self,sent):
        sent=sent.lower()
        from nltk_lite import tokenize
        word_l=list(tokenize.regexp(sent,r'\w+|[^\w\s]+'))
        if self.stem:
            p=PorterStemmer()
            word_l=[p.stemWord(w) for w in word_l]
        return word_l
    def sentence_dist(self,sent):  #get sentence word count
```

```
        import re
#        matchstr=re.compile(
#            r'<GENE_PROTEIN id="('+gname+r')">.*?(</GENE_PROTEIN>)')
        matchstr=re.compile(
            r'<GENE_PROTEIN id="MGI:(\d)+">.*?(</GENE_PROTEIN>)')
        sent=matchstr.sub(r'',sent)
        word_lst=self.words(sent)
        sent_dic={}
        for word in word_lst:
            if self.stopword.get(word,0) or len(word)<self.min_word or
word.isdigit() or not word.isalnum():
                continue
            else:
                sent_dic[word]=sent_dic.get(word,0)+1
        return sent_dic
    def gene_dist(self,g,s):  # add sentence word count s to g word count g
        for k in s.keys():
            g[k]=g.get(k,0)+s[k]
        return g
    def mesh(self, sent_dic, meshterms,subtree):  #process mesh terms and add
to sentence word dict.
        no_list=open('static/mesh.nolist','r').readlines()
        no_list=[s.strip() for s in no_list]
        no_list=dict(zip(no_list,[1]*len(no_list)))

        terms=meshterms.split('|')
        for t in terms:
            meshhead=t.split('/')[0]

            if '*' in meshhead:
                meshhead=meshhead[1:]
      #      if meshhead not in self.mesh_tree: print meshhead
            meshcode=self.mesh_tree.get(meshhead,'Z')
            if meshcode[0] in subtree and len(meshcode.split('.'))>3:
                key1=str(meshhead.lower())
                if not no_list.has_key(key1):
                  sent_dic[key1]=sent_dic.get(key1,0)+1

        return sent_dic
```

```
#####  GOTerm.py
#####  get GO terms

def build_MGI_GO_dict():
  mgi_go={}
  for line in open('static/MGI_GO.txt','r').readlines():
    mgi,go=line.strip().split('\t')
    terms=go.split('|')
    terms=[t for t in terms if not 'unknown' in t]
    mgi_go[mgi]=terms
  return mgi_go

def get_go_term(gene_id,factor):
  mgi_go=build_MGI_GO_dict()
  go_term={}
  for g in gene_id.keys():
    terms={}
    for id in gene_id[g]:
      if mgi_go.has_key(id):
        for go in mgi_go[id]:
          terms[go]=terms.get(go,0)+factor
    go_term[g]=terms
  return go_term
```

```
###### mesh_text.py
###### process mesh and text terms


from text_proc import *
import os,re
conn=DB_conn('static/mice_gene')

def get_mesh(gene_list,mesh_subtree,maxAbs,minWordLen,stem):
      sql="""select distinct a.pmid, a.sent_pos from gene_pmid_pos a, pmid_dp b
    where MGI_gene='%s' and a.pmid=b.pmid order by b.dp desc limit '%i';"""
      sql2="""select mesh from pmid_dp where pmid='%s';"""
      proc=word_proc(minWordLen,stem)
      gene_pmid={}
      gene_words={}
      for g_name in gene_list.keys():
          for g_id in gene_list[g_name]:
                rows=conn.query(sql%(g_id, maxAbs))
                if not rows:
                  #  print 'no record for',g_name
                    continue
                pmid_list=[r[0] for r in rows]
        # store pmid and sent # list for each gene
                p_ls=gene_pmid.get(g_name,[])
                p_ls.extend(rows)
                gene_pmid[g_name]=p_ls
                for p in pmid_list:
                    meshwords=conn.query(sql2%(p))[0][0]
                    if meshwords:

gene_words[g_name]=proc.mesh(gene_words.get(g_name,{}),meshwords,mesh_subtree)
      return gene_words, gene_pmid

def get_geneWord(gene_pmid,minWordLen,stem):
    processor=word_proc(minWordLen,stem)
    sql="""select sentence from pmid_sentence where pmid='%s' and
sent_pos='%i';"""
    gene_word={}
    for g_name in gene_pmid.keys():
      for s in gene_pmid[g_name]:
        sent=conn.query(sql%(s[0],s[1]))[0][0]
        sent_word=processor.sentence_dist(sent)

gene_word[g_name]=processor.gene_dist(gene_word.get(g_name,{}),sent_word)
    return gene_word
```

```
##### option_class.py
##### options for the application

class options:
    def
__init__(self,mesh,meshSubtree,min_word_len,max_abstract,stem,wordFeature,GO,go
Factor):
      self.mesh=mesh
      self.meshSub=meshSubtree
      self.minWordLen=min_word_len
      self.maxAbs=max_abstract
      self.stem=stem
      self.wordFeature=wordFeature
      self.GO=GO
      self.goFactor=goFactor
```

```
###### application home page
###### mainhtml.py

mainhtml="""
<h1 align=center style='text-align:center'><i><span style='font-size:24.0pt;
color:#FFCC00'>Mouse</span></i><i><span style='font-size:24.0pt'> <span
style='color:#009900'>Gene</span> <span style='color:#3366FF'>Information
</span><span
style='color:lime'>Summary</span> from <span style='color:#CC33CC'>Medline
</span><span
style='color:#CC0000'>Abstracts</span></span><o:p></o:p></i></h1>

<p class=MsoNormal align=center style='text-align:center'><o:p>&nbsp;</o:p></p>

<p class=MsoBodyText align=center style='margin-top:0in;margin-right:1.25in;
margin-bottom:0in;margin-left:.5in;margin-bottom:.0001pt;text-
align:center'><span
style='font-size:12.0pt;font-family:Arial'>This prototype system explores and
selects
informative sentences for genes from Medline abstracts of 1994 to 2003. After
you enter the names of genes, the system will build
functional clusters of the genes, and provide top informative sentences for
review. <o:p></o:p></span></p>

<br>
<table>
<COLGROUP>
   <COL width="10%">
   <COL width="5%">
   <COL width="70%">
   <COL width="15%">
<tr>
<td  align='left' valign='top'>
<span style='font-size:18.0pt' style='color:#009900'>
<B><A HREF="index.py"> Home Page </A></B></span><br><p>
<span style='font-size:18.0pt' style='color:#009900'>
Gene Set Information</span><br><p>
<span style='font-size:18.0pt' style='color:#009900'>
Gene Clusters</span><br><p>
<span style='font-size:18.0pt' style='color:#009900'>
Gene Sentences</span><br>
</td>
<td>
</td>
<td>
<form action="index2.py" method=post
enctype="application/x-www-form-urlencoded">

<p class=MsoNormal>&nbsp;&nbsp;<h3>Please enter gene names and
their expression levels in fold change (separated by <b>colon ':'</b> or
<b>tab</b>) one per line in the following box.
Examples:</h3><br>
&nbsp;&nbsp;NR1:-13<br>
&nbsp;&nbsp;IL6:9</p>

<p><span style='mso-spacerun:yes'></span><TEXTAREA ROWS="14" COLS="74"
NAME="names[]"></TEXTAREA><span
style='mso-spacerun:yes'></span><o:p></o:p></p>
```

```
<p><INPUT TYPE="submit"  VALUE="Send"
><span
style='mso-spacerun:yes'><INPUT TYPE="reset"><o:p></o:p></span></p>

</form>

<p><h3>Or you can upload a text file with gene names
and their expression levels (<b>tab</b> delimited) one entry per
line.</h3><o:p></o:p></p>

<p>
<FORM ENCTYPE="multipart/form-data" ACTION="index.py" METHOD=POST>
<input name=filename type=file size=50>

<span style='mso-spacerun:yes'> </span></span><span
style='mso-spacerun:yes'></span><span style='mso-spacerun:yes'><INPUT
TYPE="submit" VALUE="Submit file"
></span>
</FORM>
<a href="../sampleData/nikki_RMA.txt" target=new>nikki_RMA.txt</a><br>
<a href="../sampleData/nikki_PDNN.txt" target=new>nikki_PDNN.txt</a>

</td>
<td align='center' valign='bottom'>
<FORM ENCTYPE="multipart/form-data" ACTION="option.py" target='new'
METHOD=POST>
<span style='color:navy'>
<INPUT TYPE="submit" VALUE="Advanced Options"
></span>
</FORM>
<a href="../sampleData/drd2.txt" target=new>Look at a sample file</a>
</table>
  """


errPage1="""
<h1 align=center style='text-align:center'><i><span style='font-size:24.0pt;
color:#FFCC00'>Mouse</span></i><i><span style='font-size:24.0pt'> <span
style='color:#009900'>Gene</span> <span style='color:#3366FF'>Information
</span><span
style='color:lime'>Summary</span> from <span style='color:#CC33CC'>Medline
</span><span
style='color:#CC0000'>Abstracts</span></span><o:p></o:p></i></h1><p>
<p>
<table>
<COLGROUP>
   <COL width="10%">
   <COL width="5%">
   <COL width="85%">
<tr>
<td background='raindrop.jpg' align='center' valign='top'>
<span style='font-size:18.0pt' style='color:#009900'>
<B><A HREF="index.py"> Home Page </A></B></span><br><p>
<span style='font-size:18.0pt' style='color:#009900'>
Gene Set Information</span><br><p>
<span style='font-size:18.0pt' style='color:#009900'>
Gene Clusters</span><br><p>
<span style='font-size:18.0pt' style='color:#009900'>
Gene Sentences</span><br>
```

```
</td>
<td></td>
<td>
<b> We detect an err. No gene name entered or incorrect separation of gene name
and fold change.</b>

      <FORM ENCTYPE="multipart/form-data" ACTION="index.py" METHOD=POST><p>
<INPUT TYPE="submit" VALUE="Go Back to Re-enter."
></span><span style='color:#333399'><o:p></o:p></span></p>
</FORM>
</td>
</table>
"""

errPage2="""
<h1 align=center style='text-align:center'><i><span style='font-size:24.0pt;
color:#FFCC00'>Mouse</span></i><i><span style='font-size:24.0pt'> <span
style='color:#009900'>Gene</span> <span style='color:#3366FF'>Information
</span><span
style='color:lime'>Summary</span> from <span style='color:#CC33CC'>Medline
</span><span
style='color:#CC0000'>Abstracts</span></span><o:p></o:p></i></h1><p>
<p>
<table>
<COLGROUP>
   <COL width="10%">
   <COL width="5%">
   <COL width="85%">
<tr>
<td background='raindrop.jpg' align='center' valign='top'>
<span style='font-size:18.0pt' style='color:#009900'>
<B><A HREF="index.py"> Home Page </A></B></span><br><p>
<span style='font-size:18.0pt' style='color:#009900'>
Gene Set Information</span><br><p>
<span style='font-size:18.0pt' style='color:#009900'>
Gene Clusters</span><br><p>
<span style='font-size:18.0pt' style='color:#009900'>
Gene Sentences</span><br>
</td>
<td>
</td>
<td>
<b> We detect an err. No file name entered.</b>

      <FORM ENCTYPE="multipart/form-data" ACTION="index.py" METHOD=POST><p>
<INPUT TYPE="submit" VALUE="Go Back to Re-enter."
></span><span style='color:#333399'><o:p></o:p></span></p>
</FORM>
</td>
</table>
"""
```

```
###### second.html

<h1 align=center style='text-align:center'><i><span style='font-size:24.0pt;
color:#FFCC00'>Mouse</span></i><i><span style='font-size:24.0pt'> <span
style='color:#009900'>Gene</span> <span style='color:#3366FF'>Information
</span><span
style='color:lime'>Summary</span> from <span style='color:#CC33CC'>Medline
</span><span
style='color:#CC0000'>Abstracts</span></span><o:p></o:p></i></h1><p>
<br>
<p>

<table>
<COLGROUP>
   <COL width="10%">
   <COL width="5%">
   <COL width="85%">
<tr>
<td align='left' valign='top'>
<span style='font-size:20.0pt' style='color:#009900'>
<A HREF="index.py"> Home Page </A></span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
<b>Gene Set Information</span></b><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
Gene Clusters</span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
Gene Sentences</span><br>
</td>
<td>
</td>
<td>
<h3>This system works at gene name level. If your data set has transcript level
expression profile, it may result in one gene name representing different
transcripts. </h3><br>
<b>We have checked the input gene names for duplicates.<p>
#if $duplicate:
The following duplicates are found:<br>
#for $i in $duplicate:
    <b>$i</b>
    <br>
#end for
#else:
No duplicates are found.
#end if

<p>
#if $no_record:
<b>The following genes do not have MGI_IDs:</b><br>
#for $i in $no_record:
    <b><span style='color:$genes[$i][1]'>$i:$genes[$i][0]</span><br>
#end for
#end if

<p>
#if $count>1
<table>
<tr>
<td>
```

```
<b>Press Continue to cluster these $count genes or use Evaluation Version to
evaluate the tool:</b><p>
</tr>
<tr>
<td>
<FORM ENCTYPE="multipart/form-data" ACTION="term_modeler.py" METHOD=POST>
<INPUT TYPE="submit" VALUE="Continue">
</FORM>

</tr>
</table>
<p>
<b>These are the MGI_IDs associated with these genes that are used to retrieve
information:</b><br>
<table>
#for $i in $gene_id:
  <tr>
  <td>
    <b><span style='color:$genes[$i][1]'>$i:$genes[$i][0]</span></td>
  <td>
    #for $j in $gene_id[$i]:
      <A
HREF="http://www.informatics.jax.org/searches/accession_report.cgi?id=$j"
target='new'>$j</A>
    #end for
      </b>
  </td>
#end for
</table>
<p>
<p>
<b>Use the BACK button in your browser to go back and modify input
list.</b><br>
#else
<b>Cannot find any information on genes in your dataset. Use the BACK button in
your browser to go back and modify input list.</b><br>
#end if
</td>
</table>
```

```
###### third.html

<h1 align=center style='text-align:center'><i><span style='font-size:24.0pt;
color:#FFCC00'>Mouse</span></i><i><span style='font-size:24.0pt'> <span
style='color:#009900'>Gene</span> <span style='color:#3366FF'>Information
</span><span
style='color:lime'>Summary</span> from <span style='color:#CC33CC'>Medline
</span><span
style='color:#CC0000'>Abstracts</span></span><o:p></o:p></i></h1><p><br>

<table>
<COLGROUP>
   <COL width="10%">
   <COL width="5%">
   <COL width="85%">
<tr>
<td align='left' valign='top'>
<span style='font-size:20.0pt' style='color:#009900'>
<A HREF="index.py"> Home Page </A></span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
<b>Gene Set Information</span></b><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
Gene Clusters</span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
Gene Sentences</span><br>
</td>
<td>
</td>
<td>
<h3>We have gathered GO terms, MeSH terms and MedLine abstracts information
associated with the genes.</h3>
<p>
#if $no_record:
<b>We can not find information in our database for the following genes:</b><br>
#for $g in $no_record:
    <b><span style='color:$genes[$g][1]'><A
HREF="http://www.informatics.jax.org/searches/accession_report.cgi?id=
#for $i in $id[$g]:
  $i,
#end for
">$g:$genes[$g][0]</A><br>
#end for
#end if

<p>
<b>Press continue to view clustering results or use BACK in your browser to go
back and modify input list.</b><br>
<FORM ENCTYPE="multipart/form-data" ACTION="cluster.py" METHOD=POST>
<input name=out type=hidden value="$out">
<input name=cluster type=hidden value="$cluster">
<input name=genename type=hidden value="$genename">
<span style='color:navy'></span><INPUT TYPE="submit" VALUE="Continue">
</FORM>

</td>
</table>
```

```
####### fourth.html


<h1 align=center style='text-align:center'><i><span style='font-size:24.0pt;
color:#FFCC00'>Mouse</span></i><i><span style='font-size:24.0pt'> <span
style='color:#009900'>Gene</span> <span style='color:#3366FF'>Information
</span><span
style='color:lime'>Summary</span> from <span style='color:#CC33CC'>Medline
</span><span
style='color:#CC0000'>Abstracts</span></span><o:p></o:p></i></h1><p>
<br>
<p>

<table>
<COLGROUP>
   <COL width="10%">
   <COL width="5%">
   <COL width="85%">
<tr>
<td align='left' valign='top'>
<span style='font-size:20.0pt' style='color:#009900'>
<A HREF="index.py"> Home Page </A></span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
Gene Set Information</span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
<b>Gene Clusters</b></span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
Gene Sentences</span><br>
</td>
<td>
</td>
<td>
<table border="1">

   <th bgcolor=#FFCC00>Cluster and Summary Keywords</th>
   <th bgcolor=#FFCC00>Genes in the Cluster and Their Top Five Associated
Terms</th>
#for $c in $sort_list:
  <tr>
  <td>
  <b>Cluster $c:</b>
  <br>

    #for $d in $desc_word[$c]:
      #if $desc_word[$c].index($d)>4:
       #break
      #end if
      $d #slurp
      <br>
    #end for

  </td>
  <td>
  #for $g in $cluster_gene[$c]:

    <b><span style='color:$genes[$g][1]'><A
HREF="http://www.informatics.jax.org/searches/accession_report.cgi?id=
#for $i in $id[$g]:
  $i,
#end for
">$g:$genes[$g][0]</A>
    :
    #for $term in $topTerms[$g]:
      $term;
    #end for
    </span></b><br>
    #if $g in $gene_pmid:
    <FORM ENCTYPE="multipart/form-data" ACTION="sentence.py" METHOD=POST
target='new'>
    Enter query terms: <input name=query type=text>
    <input name=desc_word type=hidden value="$sub_desc[$c]">
    <input name=gene type=hidden value="$g">
    <span style='mso-spacerun:yes'>     </span></span><span
style='color:navy'><span
    style='mso-spacerun:yes'>     </span><span style='mso-
spacerun:yes'> </span><INPUT TYPE="submit"    VALUE="Get
Sentences"></span><span style='color:#333399'><o:p></o:p></span></p>
    </FORM>
    #else:
    <FORM ENCTYPE="multipart/form-data"
ACTION="http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?CMD=search&DB=pubmed&term
=$g" METHOD=POST target='new'>
    No sentences in database. <INPUT TYPE="submit"    VALUE="Search
PubMed"></form>
    #end if
  #end for
  </td>
  <p>
#end for
</td>
</table>
```

```
##### fifth.html


<h1 align=center style='text-align:center'><i><span style='font-size:24.0pt;
color:#FFCC00'>Mouse</span></i><i><span style='font-size:24.0pt'> <span
style='color:#009900'>Gene</span> <span style='color:#3366FF'>Information
</span><span
style='color:lime'>Summary</span> from <span style='color:#CC33CC'>Medline
</span><span
style='color:#CC0000'>Abstracts</span></span><o:p></o:p></i></h1><p>
<br><p>

<table>
<COLGROUP>
   <COL width="10%">
   <COL width="5%">
   <COL width="85%">
<tr>
<td align='left' valign='top'>
<span style='font-size:20.0pt' style='color:#009900'>
<A HREF="index.py"> Home Page </A></span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
Gene Set Information</span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
Gene Clusters</b></span><p>
<span style='font-size:20.0pt' style='color:#009900'>
<b>Gene Sentences<br></span><br>
</td>
<td>
</td>
<td>
<h3>Here are the sentences for gene <A
HREF="http://www.informatics.jax.org/searches/accession_report.cgi?id=
#for $i in $id:
  $i,
#end for
">$gene</A></h3>

<table >
<COLGROUP>
   <COL width="75%">
   <COL width="25%">
<tr>
<td>
#for $s in $sort_score:
Score=
  $s[1]&nbsp;&nbsp;&nbsp;
<A
HREF="http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&list
_uids=$s[0][0]&dopt=Abstract" target='new'>
  Go to Abstract</A>
  <br>
  $sentence[s[0]]
  <br><p>
#end for
  </td>
<td align='center' valign='top'>
<FORM ENCTYPE="multipart/form-data" ACTION="sortByAbstract.py" METHOD=POST>
<input name=gene type=hidden value="$gene">
```

```
<span style='mso-spacerun:yes'> </span>
<span style='mso-spacerun:yes'> </span>
<span style='color:navy'>
<INPUT TYPE="submit" VALUE="Arrange sentences by abstract"
></span>
</FORM>


</td>
<p>
</tr>

</table>
```

```
##### sixth.html


<h1 align=center style='text-align:center'><i><span style='font-size:24.0pt;
color:#FFCC00'>Mouse</span></i><i><span style='font-size:24.0pt'> <span
style='color:#009900'>Gene</span> <span style='color:#3366FF'>Information
</span><span
style='color:lime'>Summary</span> from <span style='color:#CC33CC'>Medline
</span><span
style='color:#CC0000'>Abstracts</span></span><o:p></o:p></i></h1><p>
<p><br>
<p>

<table>
<COLGROUP>
   <COL width="10%">
   <COL width="5%">
   <COL width="85%">
<tr>
<td align='left' valign='top'>
<span style='font-size:20.0pt' style='color:#009900'>
<A HREF="index.py"> Home Page </A></span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
Gene Set Information</span><br><p>
<span style='font-size:20.0pt' style='color:#009900'>
Gene Clusters</b></span><p>
<span style='font-size:20.0pt' style='color:#009900'>
<b>Gene Sentences<br></span><br>
</td>
<td>
</td>
<td>
<h3>Here are the sentences for gene <A
HREF="http://www.informatics.jax.org/searches/accession_report.cgi?id=
#for $i in $id:
  $i,
#end for
">$gene</A> arranged by abstract:</h3>

<table >
<COLGROUP>
   <COL width="75%">
   <COL width="25%">
<tr>
<td>
#for $ab in $sortAbs:
  #set flag=1
  #for $st in $ab:
    #if $flag:
      Score=$st[1] &nbsp;&nbsp;&nbsp;
      <A
HREF="http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&list
_uids=$st[0][0]&dopt=Abstract" target='new'>Go to Abstract</A>
      <br>
      $sentence[$st[0]]
      <UL>
      #set flag=0
    #else:
      <LI><b>Score=</b>
      $st[1]
      &nbsp;&nbsp;&nbsp;
      <br>
      $sentence[$st[0]]
    #end if
  #end for
  </UL>
  <p><p>

#end for
</td>
<td>
</td>
</tr>
</table>
</td>
</table>
```