# A Framework for Hypothesis Generation and Knowledge Discovery in Medical Domain Literature

Raheleh Shiri-Varnaamkhaasti

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

at

Concordia University

Montreal, Quebec, Canada

December 2007

*Your file Votre référence*
*ISBN: 978-0-494-40953-4*
*Our file Notre référence*
*ISBN: 978-0-494-40953-4*

# ABSTRACT

A Framework for Hypothesis Generation and Knowledge Discovery in Medical Domain Literature

Raheleh Shiri-Varnaamkhaasti

The vast amount of information in medical domain and health sciences collected in databases such as MEDLINE is growing rapidly. There has been increased interest in discovering the so-called new public knowledge from such databases. Swanson proposed an approach, called the ABC model, for mining Undiscovered Public Knowledge (UPK) in medical literature. Since its introduction, several attempts have been made in adopting and using the ABC model. Extensibility would be a key feature making it easier to develop future extensions. Noting the increased interest in using the model, we investigate properties of a desired framework which can be easily extended and enhanced. The exploratory nature of UPK discovery requires that the data mining tools be interactive and flexible. Also, the large amount of data to be processed needs to be handled efficiently. We identify three basic requirements: flexibility, extensibility, and interactivity, and show they can be realized by taking advantage of the pipes and filters architecture. The efficiency of our framework is due to allowing concurrent execution of multiple threads, provided as an additional benefit of our architectural design. We have designed and implemented a running prototype, ExaminMED, which provides various features such as possibility of choosing filters, adding or removing terms, and comparing and combining the results of various searches. The proposed framework has essential ingredients as an effective tool for UPK discovery.

# ACKNOWLEDGEMENTS

First and fore-most, I want to thank God for giving me all that was required to complete this thesis. I would not have been able to accomplish this without his grace.

I would like to express my deep gratitude to my supervisors, Dr. T. Radhakrishnan and Dr. S. P. Mudur, for their continuous support, encouragement, guidance, and valuable advice. I learned a great deal from them. It has been a privilege to work under their supervision.

I would especially like to thank my loving parents, husband, brother, and sister for believing in me and giving me the love, motivation, encouragement, and support that I needed to complete this thesis. I am grateful for all their sacrifices. I thank them from the bottom of my heart!

Last but not least, I thank my former and present lab-mates for the useful discussions we had in the lab and also for their help in different development stages of this thesis.

# TABLE OF CONTENTS

LIST OF FIGURES ..... vii
LIST OF TABLES ..... viii
CHAPTER 1: INTRODUCTION ..... 1
1.1 Motivating Example ..... 3
1.2 Problem Statement ..... 4
1.3 Thesis Contributions ..... 7
1.4 Organization of the Thesis ..... 9
CHAPTER 2: BACKGROUND AND RELATED WORK ..... 10
2.1 Knowledge Discovery and Text Mining ..... 10
2.2 MEDLINE and BioText ..... 12
2.3 Pipes and Filters Software Architecture ..... 13
2.4 MeSH and UMLS Ontologies ..... 14
2.4.1 MeSH Ontology ..... 15
2.4.2 UMLS Ontology ..... 17
2.5 Swanson's ABC Model ..... 22
2.5.1 Weeber's DAD-System ..... 26
2.5.2 Srinivasan's Profile-based Approach ..... 27
2.5.3 Yoo's Bio-SbKDS ..... 29
2.6 Possible Improvements ..... 31
CHAPTER 3: OUR PROPOSED FRAMEWORK ..... 33
3.1 Framework Description ..... 33
3.2 System Architecture ..... 43
3.2.1 Access and Transformation of MEDLINE ..... 45
3.2.2 Relational Database Management System (RDBMS) ..... 46
3.2.3 Unified Medical Language System Knowledge Source Server (UMLSKS) ... 51
3.2.4 The Main Module ..... 52
3.2.5 Simple Graphical User Interface ..... 52
CHAPTER 4: FEATURES AND ADVANTAGES OF ExaminMED ..... 55
4.1 User Interaction Features and Advantages ..... 57
4.1.1 Flexibility in Selecting Desired Filters ..... 58
4.1.2 Refining Returned Results ..... 60
4.1.3 Saving and Combining Search Results using SQL ..... 63
4.1.4 Viewing Corresponding Terms ..... 69
4.1.5 Sorting Terms ..... 71
4.2 Computational Features and Advantages ..... 72
4.2.1 Flexibility Due to Pipes and Filters ..... 73
4.2.2 Concurrency Support ..... 74
4.2.3 Use of Database Technology to Store and Access MEDLINE ..... 74
4.3 Non-Functional Features and Advantages ..... 75
4.3.1 Extensibility ..... 75
4.3.2 Usability ..... 76
4.3.3 Scalability ..... 77
CHAPTER 5: IMPLEMENTATION AND EVALUATION ..... 78

5.1 Implementation Details ..... 78
5.2 Comparison of Various Versions of the Software ..... 80
5.3 Correctness and Usefulness ..... 87
5.3.1 Correctness ..... 87
5.3.2 Evaluation of Usefulness ..... 92
CHAPTER 6: CONCLUSION AND FUTURE WORK ..... 95
6.1 Summary and Conclusion ..... 95
6.1.1 Interactive, Flexible, Extensible, and Concurrent UPK Discovery ..... 95
6.2 Future Work ..... 99
6.2.1 Sharing Results among Experts ..... 99
6.2.2 Using Specialized Ontology instead of MeSH and UMLS ..... 100
6.2.3 Parallel Processing ..... 100
LIST OF REFERENCES ..... 102

# LIST OF FIGURES

Figure 2-1. Tree structure of Cardiovascular Diseases ..... 16
Figure 2-2. Semantic types in UMLS ..... 19
Figure 2-3. "Biologic Function" Semantic Type Hierarchy ..... 21
Figure 3-1. Arrangement of pipes when semantic type filtering is included ..... 38
Figure 3-2. Arrangement of pipes when semantic type filtering is excluded ..... 39
Figure 3-3. Process Flow Diagram using Semantic Filtering ..... 42
Figure 3-4. Process Flow Diagram not using Semantic Filtering ..... 43
Figure 3-5. ExaminMED System Architecture ..... 44
Figure 3-6. Database Table Names and Their Relationships ..... 48
Figure 3-7. Graphical User Interface ..... 54
Figure 4-1. Results Displayed for the Raynaud Disease Example ..... 58
Figure 4-2. Semantic Filter not Selected ..... 60
Figure 4-3. Removing the B-term "Hypertension" ..... 62
Figure 4-4. Combining Results Window ..... 64
Figure 4-5. Combining Results Query ..... 65
Figure 4-6. Results of Query: "Migraine Disorder and Raynaud Disease" ..... 67
Figure 4-7. Pipe Setup for Query "C1 AND C2" ..... 68
Figure 4-8. Pipe Setup for Complex Combinations ..... 69
Figure 4-9. B-terms Corresponding to Cortisone ..... 70
Figure 4-10. Sorted A-terms ..... 72
Figure 5-1. GUI of Versions 1 and 2 of the Prototype ..... 83

# LIST OF TABLES

Table 2-1. ARROWSMITH's Sample Output for the Migraine-Magnesium Example ... 25
Table 3-1. Conceptual Database Design ... 49
Table 5-1. Classes and Their Main Functionalities ... 80
Table 5-2. Runtimes for Different Versions of our Implementation ... 86
Table 5-3. Semantic Types for B-terms and A-terms Produced by our Prototype ... 89
Table 5-4. Number of Co-occurrence Entries of Top-twenty A-terms with Raynaud Disease in PubMed ... 91

# CHAPTER 1

# INTRODUCTION

The amount of knowledge in different scientific areas is continually growing at a fast pace. For example, in the areas of life sciences and biomedicine, biomedical bibliographic information has been collected for decades and recorded as texts in databases such as MEDLINE, PubMed Central, and BioMed Central [22]. Presently, MEDLINE is the most comprehensive online information system in these areas, with over 16 million records covering the fields of medicine, nursing, pharmacy, dentistry, veterinary medicine, and health care. It also contains information on much of the literature in biology and biochemistry, and even fields that have no direct medical connection, such as molecular evolution. MEDLINE is compiled by the U.S. National Library of Medicine (NLM), and is freely available on the Internet and is searchable, e.g., through PubMed. About 10,000 articles are added to the database per week. Considering the huge size of the database, it is but natural to assume that it contains unnoticed and hidden information, which may possibly be discovered using suitable data mining tools and techniques.

The concept of Undiscovered Public Knowledge (UPK) was first introduced by Don Swanson about 20 years ago [17]. The idea is that undiscovered knowledge can be generated and revealed by mining knowledge bases. In the biomedical domain, the amount of available information is so vast that the domain tends to divide itself into multiple specialties. The inevitable result of this division is the mutual isolation of the different specialties. Therefore, it is possible that by considering the different mutually

isolated sub-areas together, one may discover new knowledge, UPK, which was never documented, and hence supposedly not known before. The ABC model, proposed by Swanson, is a well established approach to hypothesis generation and UPK discovery and is described as follows. Given a concept C, suppose there is a collection of bibliographic literature indicating that some other concepts B are related to C. Also suppose there is a different collection, which is in complete isolation from the first one, containing information that relates the B concepts to yet another set of concepts, say A. Further suppose there is no literature which records or claims any direct correlation between the given C and A concepts. It is then possible to hypothesize that there is an unknown connection between C and A, through the B concept. In other words, since the medical literature confirms a C-B connection and a B-A connection, a clinical researcher may suspect an hither-to unnoticed C-A connection which can be validated or rejected after investigation and/or performing further clinical tests. Finding relationships between such C, B, and A concepts may not be easy for humans noting the fact that the size of the knowledge base is huge. With the help of a proper software tool, however, the B and A concepts for a given C concept can be identified and presented to the user for further exploration and insight. The design and development of such software forms the main focus of our investigation in this research. Once these concepts are identified, the user may hypothesize a hidden connection using his/her domain knowledge.

Many researchers have tried to replicate, improve, and extend Swanson's ABC model for UPK discovery [16, 20, 22]. While specific software programs have been individually developed in these endeavors, they all have their limitations for wider application.

In this work, we propose a flexible, extensible, and interactive framework which applies the ABC model and supports hypothesis generation and knowledge discovery. Considering the purpose for which the software is built, the framework should fulfill certain requirements. We next illustrate the ideas through an example and further motivate our work. This is followed by a formal problem statement. Thesis contributions and the organization of the thesis document are presented at the end of this chapter.

## *1.1 Motivating Example*

As mentioned above, MEDLINE is a very large database with millions of articles and bibliographic information in biomedical and health sciences. It is too difficult a task, in general, for humans to discover new knowledge that possibly exists in these articles for the purpose of UPK discovery. Let us suppose, for example, that a clinical researcher is interested in investigating about *Raynaud Disease*. Searching in PubMed using the keyword *Raynaud Disease* returns 5152 articles, 3537 of which contain this term as a major heading. Reviewing all these articles in hope of discovering new knowledge hidden in the vast amount of information can be hectic, time-consuming, and perhaps impossible for humans due to restrictions such as time constraints and large search space. Now, suppose the researcher wishes to identify articles that are "connected," for having some information on *Raynaud Disease,* with that of a non-communicating set of articles. This task is even more complicated than just searching one set of articles. It is this task of looking for new, undiscovered knowledge which can be made feasible and can be facilitated with the help of computers, as was first demonstrated using the ABC model.

The ABC model has proven to be an effective approach for searching large biomedical databases such as MEDLINE in order to discover hidden unknown knowledge. Multiple unnoticed links have been discovered through this innovative idea, such as the discovery of the connection between *Raynaud Disease* and *Dietary Fish Oils* and also the connection between *Migraine Disorders* and *Magnesium Deficiency* through various B concepts. Analytical and mining tools and techniques can be used for hypothesis generation, which are then validated or rejected through further investigation.

## *1.2 Problem Statement*

A software tool designed for hypothesis generation and knowledge discovery should have certain characteristics for being effective and useful. It should be flexible, interactive, and user friendly. It should also be able to process and present the required information in a reasonable amount of time and in an easy to understand manner.

The flexibility requirement for such a software system is to allow the user to experiment with and explore the available data through the features included in the software. Flexibility is even more appreciated when the software is interactive, since it gives users easier access to the different options and control over the system in the discovery process. Even though a computer system can help significantly in dealing with large amounts of data, we still believe that human intelligence is irreplaceable in this context. In other words, we need a software system that automates some of the steps in the hypothesis generation process, while still benefiting from the expertise, experience, and intelligence of the user. Therefore, a software developed to support hypothesis generation should be

interactive and user-centered. Since not all the users are computer experts, such software should be easy to learn and use. This is especially important for the users in our context who are medical experts and who may not afford to spend much time learning to use such software along with its various features. Having a user-friendly interface is crucial for the success of such software, hiding the complex functionalities, design, and implementation details of the system.

Since the literature in biomedical area is vast, it is important to be able to deal with this huge and rich source of knowledge in an efficient way, and provide the researchers with meaningful results in a reasonable amount of time. As some steps in the discovery process may not be completely dependent on each other and may overlap, having multiple threads of processes can significantly improve the system performance by allowing the threads to execute concurrently.

There are other related issues which are also important in our context and deserve special attention. Considering the huge data size, it is possible that the results of various search operations are also large. As the number of terms to be processed could be large, it would be beneficial to use multiple filters in order to prune the B and A concepts in Swanson's ABC model before presenting the results to the user. As a result, the user is provided with fewer but more meaningful and interesting terms to consider for further analysis and processing. For convenience, a desired system can present each result immediately after it is produced to allow early involvement of the user by browsing and absorbing the results, while the system continues computing to produce further results. This "pipelining" of returning the results to the user, as they are produced, and browsing the results by the

user is important in particular when the number of B and A terms involved for a given set of C terms is large.

Another useful feature in this context, we believe, is to store the search results in a database. This can be a basis to provide a mechanism that the user can use to combine desired search results, done through easy formulation of database queries and their evaluation.

In summary, our aim in this thesis is two-fold. On the one hand, our goal is to support medical doctors through an interactive software system in their task of creating hypotheses and potentially discovering UPK as part of their research. On the other hand, the goal is to build an extensible software framework, which can easily enable others in the future to make substantial improvements to the UPK discovery process. In such hypothesis formulation processes, medical experts use both the published medical literature and their own intuitions from clinical experience. The supporting interactive system's requirements are:

(a) Flexible at the system architecture level to provide an extensible software framework

(b) Effective user interface that is Easy to learn and Easy to use (doctors are too busy and find little time to learn technologies that do not benefit them right away)

(c) Incremental response from the system while processing a very large database so that the computational power of the computer in very large database search, and human intelligence and intuition in the formulation of hypotheses can be combined sooner than later in a cycle

(d) Flexible at the end user level to explore options

(e) Quick real time response while working with MEDLINE-like database for trying out different alternatives

(f) Ease with which the researchers can save intermediate outcomes and resume later and/or share their findings easily with fellow researchers.

As the above scope is too large, some of these requirements, for example e and f are not directly addressed but suggested as future work in continuation of this thesis work.

Creating an extensible software framework is very useful considering the many attempts made by researchers to improve Swanson's basic ABC model in the past. A flexible and extensible framework makes it simple to add new components and elaborate further on the basic idea without needing to build a new software tool from scratch.

## *1.3 Thesis Contributions*

Due to the growing nature of the MEDLINE database, devising a software system to efficiently mine the vast amount of biomedical information for discovering new hidden knowledge is of great importance. Although using computers can help in dealing with this large amount of data, we believe that the discovery process can and should be improved by allowing human interaction, to effectively discover meaningful, previously unknown, and hidden knowledge. One can device a framework that can easily be improved and extended, while taking advantage of computer technology for the purpose of UPK discovery. We define a framework as a basic conceptual structure which proposes a solution to a complex problem. It can also be viewed as a solution methodology.

Noting the nature of this application which is exploratory, we propose a flexible, extensible, and interactive framework, which also allows concurrency as an additional feature. In order to make this possible, we first view the ABC model as the process of filtering information in the knowledge-base. While in its simplest form, this filtering can be viewed as a single pipeline of two filters parameterized by the C and B terms, it is easily generalized using the well known "pipes and filters" design pattern. The pipes and filters architecture allows us to provide a flexible system which gives the user more freedom and control while interacting with the software. This flexibility is important upon noting the exploratory nature of UPK in the vast field of medical sciences. The system incorporates semantic knowledge into the ABC model in order to identify a more meaningful set of B-terms and A-terms, related to a user input C-term. We have developed a running prototype, ExaminMED (*EX*ploring *A*nd *MIN*ing *MED*line), of the proposed framework. In the system prototype, for illustration and experimentation purposes, we have implemented four filters for pruning the B- and A-terms, two of which are co-occurrence filters and the other two are semantic filters. The pipes and filters architecture allows us to easily add new filters and/or modify existing ones, and thus provides the possibility of creating a flexible and extensible framework at the architecture level. The users may choose the desired filters, save the search results in an RDBMS, and combine the results of various searches through our simple graphical user interface. The framework also allows users to combine or eliminate filters.

## *1.4 Organization of the Thesis*

The rest of this work is organized as follows. In Chapter 2, we discuss background and related work. Chapter 3 presents our work, ExaminMED, which is an extensible framework for Swanson's ABC model. We discuss technical details of the proposed framework, including its architecture design. Chapter 4 discusses the features and functionalities of the proposed framework and its advantages. In Chapter 5, we present implementation details and experimental evaluation using a running prototype system. Our final chapter contains concluding remarks as well as possible directions for future work.

# CHAPTER 2

# BACKGROUND AND RELATED WORK

In this chapter, we provide a background and review of previous work related to the topic of this thesis. This includes review of data and text mining domains followed by a brief introduction to the MEDLINE database and a tool which transforms it from XML into a relational database management system (RDBMS). A description of pipes and filters software architecture comes after that. A brief introduction to ontologies which we will use to add semantic knowledge-based filters to aid in the hypothesis generation process then follows. We review Swanson's ABC model and its variants that are published after his first research publication in the year 1986, which provide the basis for our research and development in this work. Finally, we identify the requirements of a UPK discovery system, which we elaborate more on in Chapter 3.

## *2.1 Knowledge Discovery and Text Mining*

The amount of scientific knowledge is growing at phenomenal rate. Advances in technology have made this huge amount of data available in digital form for further analysis and processing. Data mining has attracted a lot of attention as it attempts to extract previously unknown and potentially useful information hidden in data [5]. Taking advantage of the data that is already available and analyzing and understanding it are the main purposes of data mining tools and techniques, allowing important decisions and discoveries to be made based on information-rich data.

It is important to note the difference between data mining (DM) tools and information retrieval (IR) tools. DM tools aid the user in analyzing and understanding the data, while IR tools are simply used to retrieve information that is of interest to the user as it appears in the database. In other words, DM tools are used to discover hidden patterns or knowledge not stored in the database explicitly and not known previously, while IR tools are used to find information and return to the user.

Data mining, also called knowledge discovery from databases (KDD), usually involves analyzing and extracting information from structured data, that is, data stored in databases. The biomedical literature we are dealing with in this work is usually stored in text format and such text data must be either structured or amenable for computer processing for the purposes of data mining. Data in natural language text is unstructured in general, or it is semi-structured as it contains some fields such as title, authors, publication dates, and so on. But the main components in our context of biomedical domain, such as abstracts and contents, are unstructured texts presented in some natural language, English in our case. The main challenge of text mining is to mine unstructured texts.

Traditional text mining approaches have certain limitations. One limitation is that they do not consider semantics in their mining process. For example, they consider "cancer", "tumor" and "neoplasm" to be different terms, while they are semantically related in one or more ways. This relation is not taken into account in traditional text mining algorithms [22]. Making use of domain knowledge and available ontologies is the main idea of semantic text mining. Information stored in ontologies includes semantic knowledge such as synonyms, and relationships between different concepts in a certain domain. In the

biomedical domain, there are a number of ontologies available, which we will consider in our work to improve the search performance. Examples of such ontologies include MeSH and UMLS, which are described in detail in Section 2.4.

## *2.2 MEDLINE and BioText*

The MEDLINE database we will use in order to demonstrate our work consists of a collection of medical journal papers gathered by the National Library of Medicine (NLM) and indexed using MeSH terms, which are described in section 2.4. The texts in MEDLINE journal articles are read by NLM indexers and key terms are extracted in order to provide a rough sketch of what the text is about. Indexing is a basic technique used to facilitate efficient search and mining process. The 2007 MEDLINE database, for example, consists of 538 compressed zip files named 'medline07n0001' through 'medline07n0538', each containing data in XML format. MEDLINE contains articles published between the years 1865 and 2007 and ordered by publication date [11].

Querying MEDLINE in XML format is difficult, therefore the team at California University has designed and developed a software system called BioText [15] to parse the XML files and load the data contents into DB2 -- the IBM RDBMS. Relational databases with their sophisticated storage and indexing mechanisms, and with powerful query processing and optimization components make it possible to index and query very large databases. This explains the choice of RDBMS to store and manage the converted XML-formatted MEDLINE database. The above team have designed a database schema which is publicly available from their website [3]. BioText creates the database tables

based on the schema and populates them with the data obtained by parsing MEDLINE. BioText can be used in a number of software and hardware platforms. It is available in both Perl and Java and has different versions [15]. It can load MEDLINE into IBM DB2 as well as into Oracle 9i. Its different versions can run on both Sun and Intel processors.

## *2.3 Pipes and Filters Software Architecture*

One of the aims of this Master's thesis is to create a flexible and improved software implementation of the well known ABC model of Swanson for the purposes of human centered interactive discovery of potentially new hypothesis by medical doctors. For this purpose, as already mentioned earlier, we have employed the pipes and filters software architecture. The pipes and filters software architecture is an architectural pattern which can be used in systems that process a stream of data. Its main components are *filters*, which perform some operations on the stream of incoming data, and *pipes* which direct the filtered data to the next stage filter in a sequence and finally to the output screen [4]. Each filter is a separate and distinct processing module that can refine, transform, or enrich the data it receives. It may also perform a combination of refinement, transformation, and enrichment on its input data. The filters may execute concurrently as long that they have data to process. Each filter is dependent only on the output of the previous filter. Two kinds of filters exist: *active* and *passive*. Active filters pull data from the input stream and push data onto the output stream, and run as separate processes or threads. Passive filters on the other hand need to be activated by function or procedure

calls. Pipelines are usually linear, meaning that the data is processed linearly and the pipes do not form cycles.

The pipes and filters architectural pattern is suitable for systems that are developed by several developers, decomposed into several independent processing modules, or are likely to undergo requirement changes later. We found that this architecture is suitable for our application since the required processing can be divided into independent steps. In the next chapter we will elaborate more on our work which will explain our choice of using pipes and filters in developing our prototype software framework.

## *2.4 MeSH and UMLS Ontologies*

Ontology is a data model that represents the various concepts considered relevant to a domain and the relationships that exist between those concepts [21]. The Unified Medical Language System (UMLS) and Medical Subject Headings (MeSH) are well-structured ontologies which are built for the purpose of sharing and reusing biomedical knowledge. They support ways of standardizing the meanings of medical terms and making sure everyone in the medical community has the same understanding about the meanings of, and relationships between the concepts in the domain. These ontologies are also used to allow computer assistance in biomedical research.

### 2.4.1 MeSH Ontology

MeSH is the National Library of Medicine's (NLM) controlled vocabulary thesaurus. The articles in the MEDLINE database are indexed using MeSH terms. Hence, each record in MEDLINE has certain MeSH terms associated with it giving an idea about the content of that particular record.

MeSH terms are organized into hierarchical tree structures, in which each term can appear in more than one position in the trees. In total, 15 broad subject categories are represented in the tree structures each of which are further divided into more specific sub-categories. Figure 2-1 illustrates the tree structure for *Cardiovascular Diseases*, a sub-category of which is *Raynaud Disease*. [6]

Cardiovascular Diseases [C14]
Vascular Diseases [C14.907]
Aneurysm [C14.907.055] +
Angiodysplasia [C14.907.075] +
Angiomatosis [C14.907.077] +
Angioneurotic Edema [C14.907.079]
Aortic Diseases [C14.907.109] +
Arterial Occlusive Diseases [C14.907.137] +
Arteriovenous Malformations [C14.907.150] +
Arteritis [C14.907.184] +
Capillary Leak Syndrome [C14.907.218]
Cerebrovascular Disorders [C14.907.253] +
Diabetic Angiopathies [C14.907.320] +
Embolism and Thrombosis [C14.907.355] +
Erythromelalgia [C14.907.375]
Hand-Arm Vibration Syndrome [C14.907.440]
Hemorrhoids [C14.907.449]
Hepatic Veno-Occlusive Disease [C14.907.460]
Hyperemia [C14.907.474]
Hypertension [C14.907.489] +

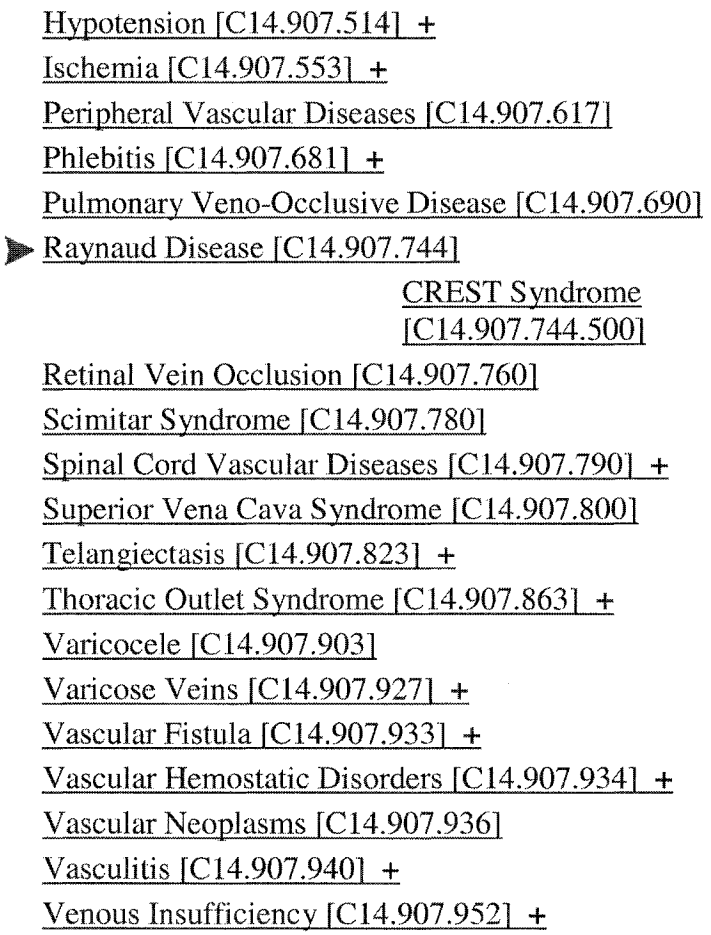


Figure 2-1. Tree structure of Cardiovascular Diseases [13]

MeSH records corresponding to research articles have three basic types: Descriptors, Qualifiers, and Supplementary Concept Records (SCRs). Descriptors are the main headings that are used to index MEDLINE records. They provide a brief description of what the article is about. Qualifiers are subheadings which are used in conjunction with Descriptors to indicate what aspect of a particular subject the article addresses. For example, "Liver" is a Descriptor and "Drug Effects" is a Qualifier. Thus, an article with the qualifier "Liver/drug effects" indicates that the theme of the article is not about liver in general, but about the effects of drugs on liver. Finally, SCRs are used to index

chemicals, drugs and other substances [10]. In our work, we deal mainly with Descriptors.

### 2.4.2 UMLS Ontology

Many other biomedical ontologies are also available, a list of which can be found in Open Biomedical Ontologies (OBO) [14]. The National Library of Medicine (NLM) attempted to build UMLS to facilitate the development of computer systems that can process biomedical literature as if they "understand" the health sciences and biomedicine language [22]. All the major biomedical ontologies or vocabularies including MeSH are integrated in UMLS. NLM has produced the UMLS knowledge source and software tools to assist software developers in building biomedical and health-related applications. The UMLS has three knowledge sources: (a) the Metathesaurus, (b) the Semantic Network, and (c) the SPECIALIST Lexicon [8]. We will briefly explain the Metathesaurus and the Semantic Network components, which we will be using in our work.

#### 2.4.2.1 UMLS Metathesaurus

The Metathesaurus is a very large database which contains information about biomedical concepts, their various names and synonyms, and relationships between the concepts. It is a multi-purpose and multi-lingual vocabulary database obtained by integrating over 100 biomedical source vocabularies. The Metathesaurus is organized according to concepts or meanings. If a concept has different contextual meanings associated with it in different

source vocabularies, all of them are included in the Metathesaurus, even if there are inconsistencies among them. In other words, the Metathesaurus does not represent a single view of the biomedical world, but multiple views as they appear in different source vocabularies, which may be useful for different tasks. Each concept in the Metathesaurus is associated with one or more '*semantic types*'. For example the semantic type "Disease or Syndrome" is assigned to the concept "Migraine Disorder."

#### 2.4.2.2 UMLS Semantic Network

The purpose of the UMLS Semantic Network is to consistently categorize the concepts in the Metathesaurus in a meaningful fashion and to determine the relationships between the concepts. The Semantic Network consists of 135 *semantic types* and 54 '*semantic relations*'. The *semantic types* in the UMLS Semantic Network are illustrated in Figure 2-2. The hierarchy illustrates the semantic types in the network as well as the 'isa' relation between them.

**Entity**
- Physical Object
  - Organism
    - Plant
      - Alga
    - Fungus
    - Virus
    - Rickettsia or Chlamydia
    - Bacterium
    - Archaeon
    - Animal
      - Invertebrate
      - Vertebrate
        - Amphibian
        - Bird
        - Fish
        - Reptile
        - Mammal
          - Human
  - Anatomical Structure
    - Embryonic Structure
    - Anatomical Abnormality
      - Congenital Abnormality
      - Acquired Abnormality
    - Fully Formed Anatomical Structure
      - Body Part, Organ, or Organ Component
      - Tissue
      - Cell
      - Cell Component
      - Gene or Genome
  - Manufactured Object
    - Medical Device
    - Research Device
    - Clinical Drug

[Entity] (continued)
- [Physical Object] (continued)
  - Substance
    - Chemical
      - Chemical Viewed Functionally
        - Pharmacologic Substance
          - Antibiotic
        - Biomedical or Dental Material
        - Biologically Active Substance
          - Neuroreactive Substance or Biogenic Amine
          - Hormone
          - Enzyme
          - Vitamin
          - Immunologic Factor
          - Receptor
        - Indicator, Reagent, or Diagnostic Aid
        - Hazardous or Poisonous Substance
      - Chemical Viewed Structurally
        - Organic Chemical
          - Nucleic Acid, Nucleoside, or Nucleotide
          - Organophosphorus Compound
          - Amino Acid, Peptide, or Protein
          - Carbohydrate
          - Lipid
            - Steroid
            - Eicosanoid
        - Inorganic Chemical
        - Element, Ion, or Isotope
    - Body Substance
    - Food

[Entity] (continued)
- Conceptual Entity
  - Idea or Concept
    - Temporal Concept
    - Qualitative Concept
    - Quantitative Concept
    - Functional Concept
      - Body System
    - Spatial Concept
      - Body Space or Junction
      - Body Location or Region
      - Molecular Sequence
        - Nucleotide Sequence
        - Amino Acid Sequence
        - Carbohydrate Sequence
      - Geographic Area
  - Finding
    - Laboratory or Test Result
    - Sign or Symptom
  - Organism Attribute
    - Clinical Attribute
  - Intellectual Product
    - Classification
    - Regulation or Law
  - Language
  - Occupation or Discipline
    - Biomedical Occupation or Discipline
  - Organization
    - Health Care Related Organization
    - Professional Society
    - Self help or Relief Organization
  - Group Attribute
  - Group
    - Professional or Occupational Group
    - Population Group
    - Family Group
    - Age Group
    - Patient or Disabled Group

**Event**
- Activity
  - Behavior
    - Social Behavior
    - Individual Behavior
  - Daily or Recreational Activity
  - Occupational Activity
    - Health Care Activity
      - Laboratory Procedure
      - Diagnostic Procedure
      - Therapeutic or Preventive Procedure
    - Research Activity
      - Molecular Biology Research Technique
    - Governmental or Regulatory Activity
    - Educational Activity
  - Machine Activity
- Phenomenon or Process
  - Human caused Phenomenon or Process
    - Environmental Effect of Humans
  - Natural Phenomenon or Process
    - Biologic Function
      - Physiologic Function
        - Organism Function
          - Mental Process
        - Organ or Tissue Function
        - Cell Function
        - Molecular Function
          - Genetic Function
      - Pathologic Function
        - Disease or Syndrome
          - Mental or Behavioral Dysfunction
          - Neoplastic Process
        - Cell or Molecular Dysfunction
        - Experimental Model of Disease
  - Injury or Poisoning

Figure 2-2. Semantic types in UMLS [7]

The nodes in the Semantic Network represent the semantic types and the edges represent the semantic relations between the types. Each semantic type is connected to the network by at least one semantic relation. The semantic types and relations are organized into a hierarchy. The semantic types may have hierarchical (e.g., "isa") or associative (e.g., "associated-with") relationships with each other. The Semantic Network covers and categorizes a broad range of domains. Some types are more general and broader than others and therefore appear at a higher level in the hierarchy. If an exact semantic type of a concept in Metathesaurus is not in the Semantic Network, it is assigned the most specific semantic type that is available. For example, the semantic type "Manufactured Object" has two children in the network, "Medical Device" and "Research Device." Manufactured devices in the Metathesaurus that are neither medical devices nor research devices are assigned "Manufactured Device" as their semantic types.

Figure 2-3 illustrates the "Biologic Function" type hierarchy in the Semantic Network and Figure 2-4 exhibits the "affects" relationship in the hierarchy [9].

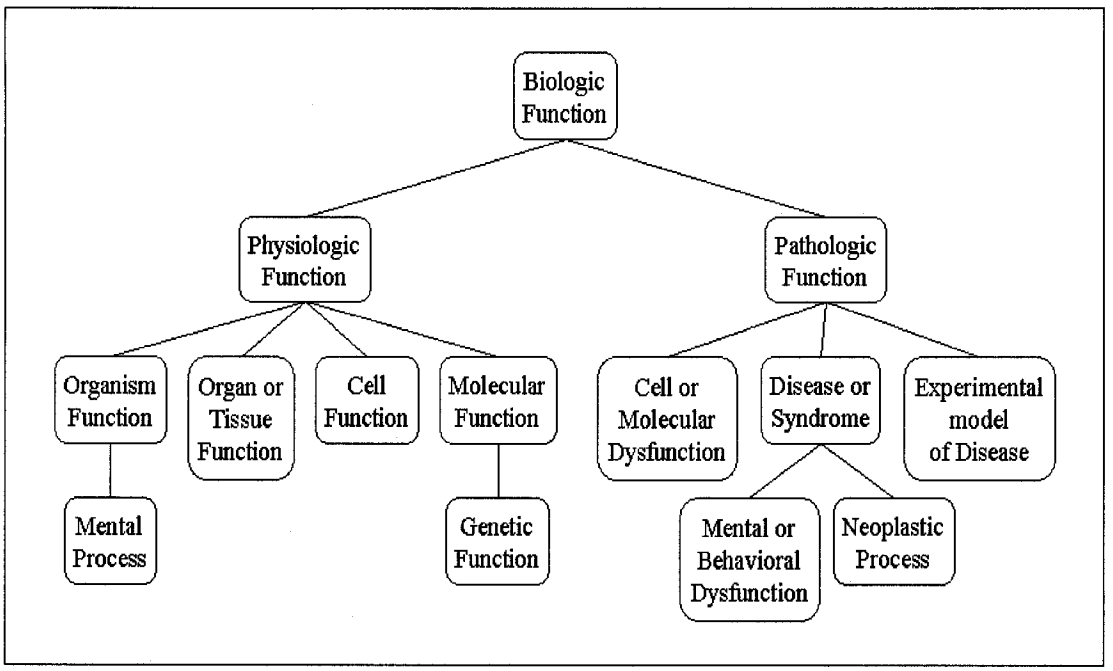


Figure 2-3. "Biologic Function" Semantic Type Hierarchy [9]

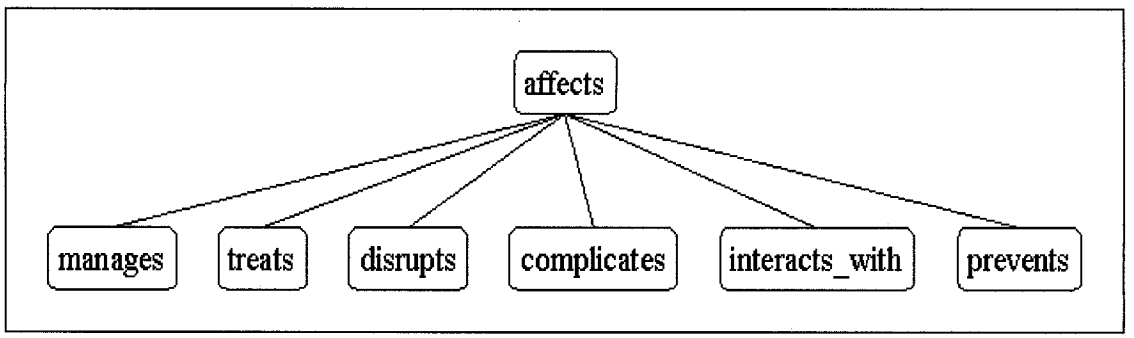


Figure 2-4. "Affects" Semantic Relation Hierarchy [9]

By making use of the UMLS Metathesaurus and Semantic Network, our goal in this thesis is to enable supplementing of the 'C', 'B', or 'A' terms in the ABC model so as to be able to extract useful and previously unknown knowledge from the biomedical literature with the assistance of the human intelligence in the interactive loop. A desired tool we plan to develop towards achieving this end goal supports experts to form and test

hypothesis while interacting with the system. This is the main idea of "discovery of UPK (unpublished public knowledge) research" to generate potentially testable hypothesis. UPK research is described next.

## *2.5 Swanson's ABC Model*

Swanson defines two bodies of knowledge to be *non-communicating* if they do not cite each other and they are not co-cited in the literature. He defines two sets of literatures to be *complimentary* if together they provide information that is not apparent from each one of them looked at separately. UPK discovery is the idea of discovering previously unknown knowledge by combining the information contained in complimentary and non-interactive literatures. This idea can be made explicit with the help of a model. The basic idea of the ABC model is that if one set of literature states that concept C is related to concept B in some way, and another set which is non-communicating with the first one claims that concept B and concept A are related, then by combining these pieces of information, one may suspect the implied connection between C and A. More specifically, let us suppose that C is a disease, B is a characteristic of C, and A is a substance that affects B. Then it is possible to hypothesize that substance A may have some bearing, e.g., cure or worsen, on disease C by affecting characteristic B. The hypothesis can then be tested in clinical labs and confirmed or rejected later. This capability to hypothesize is important upon noting the vast amount of knowledge in biomedical and health sciences domain.

During the years, numerous hidden links have been discovered using this idea and they were later tested and have led to product and service developments. Examples include the discovery of the connection between Raynaud Disease and dietary fish oil [18], and the discovery of 11 hidden links that relate Migraine and Magnesium [18].

Swanson and Smalheiser developed a software system called ARROWSMITH which serves as an aid in the UPK discovery process [19]. The basic steps of the algorithm they used in ARROWSMITH are as follows:

Algorithm- ARROWSMITH:

1. Take as input the term C in which the user is interested to discover something about.
2. Search the biomedical literature for article titles related to C, i.e., *containing* C. This identifies and generates the BC literature.
3. Use a list of stop-words generated by the user to select B-terms from the titles of the articles in the BC literature.
4. For each B term, search the biomedical literature for article titles related to B. This generates the BA literature.
5. Use a list of stop-words generated by the user to select A-terms from the article titles in the AB literature.
6. If A and C are not co-cited together in the literature, keep A.
7. Rank the A-terms based on the number of B-terms that connect them to C.

The list of stop-words may consist of biomedical terms that are too general, for example 'irradiation', as well as non-biomedical and linguistic terms such as prepositions and adverbs, which may appear in the titles of published articles. The reason why

ARROWSMITH only considers article titles as opposed to the abstract and full text is that in the biomedical domain, titles are very descriptive of the contents of the articles. A later version of ARROWSMITH uses MeSH terms in order to recognize synonyms when matching the terms, as opposed to exact-word matching. The output of ARROWSMITH is composed of two columns containing a pair of article titles at each row, assumed related, which can provide clues to new hypotheses when considered together. Table 2-1 illustrates a sample output for the migraine-magnesium example. Titles which have important keywords in common are placed together to suggest existence of a hidden link to the user. For example, a title in the migraine literature is: "Stress and Type A behavior are associated with migraine." The corresponding title in the magnesium literature is: "Stress and Type A behavior lead to body loss of magnesium." These two titles put together may suggest that magnesium-deficiency causes migraine, through the "Stress and Type A behavior" link.

| Argument 1 (migraine literature) | Argument 2 (magnesium literature) |
|---|---|
| 1. a) *Stress and Type A behavior* are associated with migraine. | 1. b *Stress and Type A behavior* lead to body loss of magnesium. |
| 2. a) Excessive *vascular tone and reactivity* may increase susceptibility to migraine. | 2. b) Magnesium can reduce *vascular tone and reactivity.* |
| 3. a) *Calcium channel blockers* can prevent migraine attacks. | 3. b) Magnesium is a natural *calcium channel blocker.* |
| 4. a) *"Spreading cortical depression"* is thought to be implicated in the early phase of a migraine attack. | 4. b) High levels of magnesium in the extracellular cerebral fluid can inhibit *spreading cortical depression* in animals. |
| 5. a) There is evidence for a connection between *epilepsy* and migraine. | 5. b) Magnesium deficiency may increase susceptibility to *epilepsy.* |
| 6. a) Migraine patients have abnormally high *platelet aggregability.* | 6. b) Magnesium can suppress *platelet aggregation.* |
| 7. a) Platelets of migraine patients are abnormally sensitive to *serotonin release.* | 7. b) Magnesium can inhibit *serotonin-induced* contractions of vascular smooth muscle. |
| 8. a) *Substance P* may be a cause of head pain in migraine. | 8. b) Magnesium can suppress *Substance P* activity. |
| 9. a) Abnormal *prostaglandin (PG) release* can aggravate vasoactivity in migraine. | 9. b) Magnesium increases *prostacyclin (PGI2) formation.* |
| 10. a) Migraine may involve sterile *inflammation* of the cerebral blood vessels. | 10. b) Magnesium has anti-*inflammatory* properties. |
| 11. a) *Cerebral hypoxia* may play a key role in migraine. | 11. b) Magnesium can protect against *brain damage from hypoxia.* |

Table 2-1. ARROWSMITH's Sample Output for the Migraine-Magnesium Example [18]

ARROWSMITH is an interesting and useful software tool which automated some of the steps involved in the discovery process. However, its major limitation is its strong dependence on human involvement. As evident from the above illustrative example, ARROWSMITH relies much on human intervention in generating the list of stop-words and pruning the B-terms in order to find meaningful and truly new links between A- and C-terms. In addition, human intervention is required in deciding on a specific category for the A-terms. In other words, the user should have a rough idea about what category

the A-terms belong to. Also, as the number of terms grows, the number of associations and relationships between them grows exponentially. Therefore, the system would return a very large number of titles in general which would be difficult for the user to go through to find truly meaningful and potentially new links. Many attempts have been made by people to overcome the shortcomings of ARROWSMITH while incorporating the basic idea of the ABC model. We trust that the problem of human intervention should be a balance with a "golden mean" between two extremes (complete automation versus complete manual approach).

### 2.5.1 Weeber's DAD-System

Using the UMLS Metathesaurus and Semantic Network can help overcome or minimize the drawbacks of ARROWSMITH. In the bio-medical domain, it is usually more meaningful to consider a combination of terms, as opposed to single terms. Using single words to represent the knowledge is an oversimplification, according to Weeber. Weeber and his team [20] developed the DAD-system (*D*isease-*A*dverse drug reaction-*D*rug, or vice versa) which uses MetaMap, a tool designed by Aronson et al. at NLM [1], to map terms in the biomedical text to UMLS concepts. Use of UMLS concepts, instead of plain text, can serve four purposes: reduce the number of B-terms and A-terms produced by the system, make sure the B's and A's are biomedical terms, exclude the dependence on the list of stop-words, and consider synonyms and textual variants in the search process [20]. After mapping the text to UMLS concepts, they use the Semantic Network as a tool for filtering the results. The filtering is performed in order to reduce the search space to a

manageable size. It should be noted that there could be a possible miss in this reduction of search space. The semantic types that are used for filtering are query-dependent and are determined by the user, based on his/her interests. At different stages of the process, different filters may be used. As an example, semantic types such as *"Biologic function"*, *"Cell function"*, *"Physiologic function"*, and *"Phenomenon or process"* are used as type restrictions to prune out B-terms in the Raynaud Disease example, while *"Vitamin"*, *"Lipid"*, and *"Element, ion, or isotope"* are types that are used to filter A-terms if the user is interested in dietary factors. The DAD-system considers two concepts to be associated if they occur together in the titles of biomedical articles. The concepts are filtered using semantic types, ranked based on concept frequency, and finally presented to the user for further analysis, exploration, and testing [20]. The presentation of the results is similar to that of ARROWSMITH. Article titles that contain A-B and B-C concepts are displayed next to each other to facilitate the hypothesis generation process. For example, the titles "Blood Viscosity and Raynaud Disease" and "Reduction in blood viscosity by eicosapentaenoic acid" are placed next to each other.

### 2.5.2 Srinivasan's Profile-based Approach

The ABC model has also been applied by Srinivasan for a similar purpose [16]. They build a profile from MEDLINE topics based on the MeSH terms for ranking the B-terms and A-terms. Srinivasan defines a profile as a set of MeSH terms representing a topic. The profiles they generate are weighted vectors of MeSH terms for various topics. The weights are computed using different weighting schemes, for example *term frequency*

multiplied by *inverse document frequency* (TF*IDF). The topics may be single words, such as *Tylenol,* or a group of words, for example *Calcium channel blockers.* Furthermore, they use the semantic knowledge incorporated in UMLS to enrich the profiles. The weights are computed within the context of a semantic type. Therefore, each profile becomes a vector of weighted MeSH vectors, one for each semantic type. Their algorithm takes three inputs: the C-term of interest, a set of semantic types used for filtering B-terms (ST-B), and a set for filtering A-terms (ST-A). Then they search PubMed for C and build its profile, limited to ST-B, and select the top ranked MeSH terms among the terms in the profile. The result gives B-terms, which are searched in PubMed and a profile is built for each one based on ST-A. Next, they sum up the weights of each MeSH term in the profiles and compute a combined profile. If the MeSH terms have not already occurred with C, they are kept as A-terms. Unlike Swanson and Weeber, Srinivasan's goal is to display, for each semantic type in ST-A, a ranked list of MeSH terms, as opposed to displaying titles and sentences next to each other. They believe that ranking the terms within each semantic type allows users to choose to keep or eliminate semantic type filtering. By looking at the first few terms in each semantic group, the user may find the top ranked MeSH terms without considering the semantic types to which they belong [16]. Also, B-terms are profiled and searched automatically, while Swanson and Weeber's approaches required human intervention in selecting B-terms. Srinivasan has also demonstrated that using the indexed MeSH terms in MEDLINE produces results competitive to the free-text based approach taken by Weeber.

### 2.5.3 Yoo's Bio-SbKDS

As stated earlier, the UMLS Semantic Network contains semantic types and semantic relations which represent relationships between the various semantic types. In the related literature we reviewed so far, only the semantic types were used if semantic knowledge was taken advantage of. Another researcher, Yoo made use of the semantic relations as well as the semantic types for filtering the B-terms and A-terms [22]. He built the software system called "Bio-SbKDS (Biomedical Semantic-based Knowledge Discovery System)" which uses the UMLS Metathesaurus and the Semantic Network in order to prune and filter the terms and suggest more meaningful A-terms to the user. Bio-SbKDS is a fully automated approach to knowledge discovery, i.e., no user interaction is required other than providing the starting input at the beginning of the process. Yoo claims that the system does not depend much on the expertise of the user and is able to generate fewer but more meaningful connections between the concepts in the biomedical domain. The semantic knowledge maintained in the Semantic Network of UMLS is used to reduce the search space while capturing meaningful yet novel connections.

For this system, the user needs to provide the following information as input: the C-term, its role, and three semantic relation filters. The output is a set of A-terms that may be linked to C-term through the various B-terms. A C-term is the term in which a user is interested in discovering something about. The role of C could be subject or object. For example suppose the user enters "Raynaud Disease" as C-term and wants to discover A-terms that "treat" or "prevent" the disease. Then the role of C would be "object" because A treats C. Therefore, A's role is "subject" and C's role is "object." Finally, the semantic relation filters include semantic relations between A and C, B and C, and also A and B.

Based on the input provided by the user, Bio-SbKDS first generates the semantic type filters for the B- and A-terms. Next, it searches MEDLINE through the PubMed interface to find articles containing the C-term. It then extracts MeSH terms in those articles and uses the semantic type filter for B-terms to prune these terms. Once all the B-terms are generated, Bio-SbKDS searches each B-term against MEDLINE, and once again the MeSH terms in those articles are extracted. A technique, proposed by Yoo, which is named "Bi-Decision Maker", is also used to filter the B- and A-terms further.

Bi-Decision Maker is used to further qualify the B- and A-terms. Suppose, for example, that C-term is "Raynaud Disease." We expect B to be symptoms of the disease and A to be something that relieves those symptoms. Therefore, B is some negative concept while A is positive. Determining whether the MeSH terms are positive or negative is achieved through analyzing the definitions of these terms. Yoo's method assigns a weight between -5 and 5 to each MeSH term based on its definition. Terms with a negative weight are considered negative and those with a positive weight are positive. For example, a B-term that was retrieved by Bio-SbKDS was "Nifedipine", which was dropped because they were looking for negative terms while Nifedipine's definition consisted of a few positive statements such as "Vasolidator", "useful", "anti-anginal", and "lower blood pressure." However, "Blood viscosity" had a negative weight and hence kept since it contained statements such as "morbidity" and "disorder" in its definition. Note that not all MeSH terms can be weighed properly because around 6% of the terms do not have definitions in NLM and also some have definitions which are neither negative nor positive.

In order to rank the A-terms, all combinations of top ranked B-terms are searched against MEDLINE. A weight is assigned to each combination, which represents the sum of

counts of elements in the combination. With this search strategy, A-terms that are related to more B-terms have higher ranks and are more strongly related to the initial C-term because they have co-occurred with more B-terms. In other words, those A-terms in the Raynaud Disease example that can relieve more than one symptom in the B list are ranked higher in the list.

To summarize, Bio-SBKDS is a fully automated approach to knowledge discovery. It does not require human intervention except at the beginning of the process where input is needed. However, we believe that human intelligence is not completely replaceable by any automated system. Computers should just be used as 'assistive tools' in the human endeavor. Human intuition in the knowledge discovery process can improve hypotheses generation by pruning automatically generated terms and exploring the results. Also, some of the steps in the algorithm can be executed in parallel, but Bio-SbKDS performs the steps sequentially. Our goal is to build a semantic-based system that aids researchers in the hypothesis generation phase and at the same time allows the researcher to intervene and contribute to the derivation of the results, while taking advantage of the pipes and filters architecture. Such a system has complementary advantages of both systems while it does not rely on humans as much as Swanson's ARROWSMITH system, and is not fully automated as Yoo's Bio-SbKDS.

## *2.6 Possible Improvements*

As mentioned earlier, there have been attempts to adopt and extend Swanson's ABC model to overcome some of its limitations or meet specific needs. There is an obvious

need for extensibility since we suspect that yet more extensions will come about in the future. Therefore, judging from past experiences where many extensions to Swanson's ABC model were proposed, we believe that creating an extensible framework is required and can indeed be useful.

Furthermore, all previous work in this endeavor has been hard-wired with little or no flexibility. Providing flexibility is essential due to the exploratory nature of UPK discovery. By providing more flexibility, we leave the user's hands free and allow him/her to be more creative in the exploration process.

In addition, since the discovery of UPK is an exploratory task, it is important to involve the user in various stages of the process and allow the user to get involved and interact with the system while benefiting from the advantages of automating some steps in the data mining task.

Finally, because of the vast amount of information that is processed, we believe that providing concurrent execution of steps wherever possible can be an additional benefit, although it is not a requirement.

# CHAPTER 3

# OUR PROPOSED FRAMEWORK

In the previous chapter we discussed Swanson's ABC model and its variations. In this chapter, we use this model as a basis to design and develop a framework, ExaminMED, for semantic mining of MEDLINE. ExaminMED is an interactive system which allows the users, essentially medical domain experts and researchers, to use their expertise and intelligence in a knowledge discovery process. More importantly, it is a framework which easily allows others to modify and potentially improve the system in the future.

This chapter is organized as follows. First, we present our framework, the underlying algorithm and its search approach. We then present the architectural design of ExaminMED, which uses the proposed algorithm, and a detailed explanation of its underlying components.

## *3.1 Framework Description*

As mentioned at the end of the previous chapter, we identified three requirements that a UPK discovery system should support. These include extensibility, flexibility, and interactivity. We believe that the pipes and filters architecture is suitable for our framework and we can benefit from a number of its advantages to fulfill the aforementioned requirements. The main advantages of using the pipes and filters architecture are easy enhancement, flexibility, efficiency, maintainability, reusability,

decoupling, and extensibility [2, 4]. It also provides concurrency, which is an additional benefit.

The following is the pseudocode of the algorithm used in ExaminMED.

**Algorithm Main** (*CTerm, CRole, CB-relationFilter, BA-relationFilter, CA-relationFilter*: *BTerm, ATerm*)

Input: CTerm: a C-term.

CRole: the role of the C term. // values = {subject, object}

CB-relationFilter, BA-relationFilter, CA-relationFilter: semantic relation filters.

Output: list of BTerms and ATerms found.

Main:

*If* semantic filtering is desired *then*

{

Get CTerm, CRole, CB-relationFilter, BA-relationFilter , CA-relationFilter;

Find semantic type of CTerm; // called ST_C

Generate possible semantic types of BTerms and ATerms; // called ST_B and ST_A

Define Pipes using Co-ocBFilter, SemBFilter, Co-ocAFilter, and SemAFilter filters;

Activate Pipes using the input CTerm, ST_B, and ST_A;

}

*else* // semantic filtering is not desired

{

Get the CTerm;

Define Pipes using Co-ocBFilter and Co-ocAFilter filters;

Activate Pipes using the input CTerm;

}

**End** Main;

**Algorithm ActivatePipes** (CTerm, ST_B, ST_A)

Input: CTerm; the user input C-term.

ST_B and ST_A: lists of semantic types generated by main algorithm.

Output: BTerm, ATerm: lists of B-terms and A-terms that match CTerm

ActivatePipes:

Activate Co-ocBFilter by feeding CTerm into the pipe;

**End** ActivatePipes;

**Algorithm Co-ocBFilter** (CTerm)

Input: CTerm; user input C-term.

Output: BTerm; B-terms that occurred with CTerm.

Co-ocBFilter:

Read CTerm;

Find synonyms of CTerm from UMLS;

Query MEDLINE for MeSH terms that co-occurred with CTerm or its synonyms; //called potentialBTerm

*For* each potentialBTerm

{

Activate semBFilter by passing potentialBTerm to it through the pipe;

}

**End** Co-ocBFilter;

**Algorithm semBFilter** (potentialBTerm, ST_B)

Input: poterntialBTerm: passed to semBFilter through co-ocBFilter.

ST_B: list of semantic types generated by main algorithm.

Output: BTerm: lists of B-terms that match CTerm and have the appropriate semantic type.

SemBFilter:

For each potentialBTerm

{

Get the semantic type of potentialBTerm from UMLS; // called semB

*If* semB exists in ST_B *then*

{

BTerm = potentialBTerm;

Display BTerm;

Activate co-ocAFilter by passing BTerm to it through the pipe;

}

}

**End** SemBFilter;

**Algorithm Co-ocAFilter** (CTerm, BTerm)

Input: CTerm; user entered C-term.

BTerm: passed to co-ocAFilter through semBFilter.

Output: ATerm; A-terms that occurred with BTerm.

Co-ocAFilter:

Read BTerm;

Find synonyms of CTerm from UMLS;

Query MEDLINE for MeSH terms that co-occurred with BTerm or its synonyms; // called potentialATerm

For each potentialATerm

{

*If* potentialATerm has not co-occurred with CTerm *then*

Activate semAFilter by passing potentialATerm to it through the pipe;

}

**End** Co-ocAFilter;

**Algorithm semAFilter** (potentialATerm, ST_A)

Input: poterntialATerm: passed to semAFilter through co-ocAFilter.

ST_A: list of semantic types generated by main algorithm.

Output: ATerm: lists of A-terms that match BTerm and have the appropriate semantic type.

```
SemAFilter:
For each potentialATerm
{
        Retrieve the semantic type of potentialATerm from UMLS; // called semA
        If semA exists in ST_A then
        {
                ATerm = potentialATerm;
                Display ATerm;
        }
}
End SemAFilter;
```

The basic search algorithm in ExaminMED is adapted from the Bio-SbKDS system which implements Swanson's ABC model. However, by benefiting from pipes and filters as our software architecture, we believe we have substantially improved the performance of the original algorithm. We add concurrency by having each filter in the pipes and filters architecture perform as a separate thread. In addition, the search results can be saved in the database for further analysis and processing. We also provide the option of combining the results of multiple searches through formulating SQL queries against the results stored in the database. Also, while Bio-SbKDS is non-interactive and presents all

search results to the user at once as a batch, we allow user interaction and provide the results as soon as they are retrieved by the system. This allows early interaction, incremental control and reflection on the part of the user. User interaction is also made possible by allowing the user to choose desired filters, e.g., the user may choose to eliminate semantic filtering if that is what suits his/her needs. This flexibility is possible due to the incorporation of the pipes and filters architecture. Other than improving the performance, using the pipes and filters architecture allows us to provide a framework that can easily be enhanced and extended. More filters can be added as needed without needing to modify the other components.

Currently there are four filters implemented in our framework: *coocBFilter*, *coocAFilter*, *semBFilter*, and *semAFilter*. The first two are co-occurrence filters for B-and A-terms while the last two are semantic filters for these terms. The pipes are set differently depending on the user's choice. If the user selects semantic filtering, all the above four filters are connected through the pipes, as illustrated in Figure 3-1.

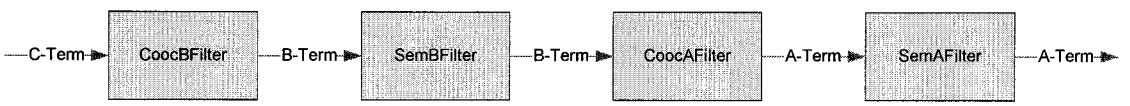


Figure 3-1. Arrangement of pipes when semantic type filtering is included

If the user chooses not to incorporate semantic filtering, only the co-occurrence filters are set through the pipes. Figure 3-2 demonstrates the arrangements of the pipes in case semantic filtering is not desired. This flexibility is a natural consequence and advantage of the architecture of our system that is based on pipes and filters. This basis provides

more opportunities for users to interact and explore, as it takes into account his/her inputs and interests. Using different filtering mechanisms may lead to formulating interesting hypotheses.

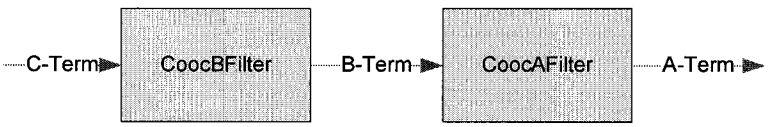


Figure 3-2. Arrangement of pipes when semantic type filtering is excluded

If semantic filtering is desired by the user, ExaminMED takes the following inputs:

1. C-term: the starting term from which the user is hoping to find unknown link(s)
2. role of the C-term: this role could be subject or object
3. The semantic relation filters: they include filters between C and B, between B and A, and between C and A

The user initiates the search process by clicking the "Search Medline" button. This establishes a connection between the system component and the UMLS in order to identify and retrieve ST_C, the semantic type of the input C-term. To determine the semantic types, ST_B, of possible B-terms, a search in the local database is performed to identify those semantic types which have at least one of the relations present in the C-B relation filter with ST_C. For example, if C has the semantic type "Disease or Syndrome" and the C-B relation filter includes the relation "process_of", and C's role is "object", then "Cell Function" is added to the ST_B list as a potential semantic type of B-terms, since the UMLS semantic network indicates that "Cell Function" is a "process_of" "Disease or Syndrome." On the other hand, if the role of C-term was assigned as "subject", then "Biologic Function," instead, would have been added to ST_B since

"Disease or Syndrome" is a "process_of" "Biologic Function." If the semantic type is "too general", it is eliminated from the ST_B list. In our implementation, the semantic types appearing in the first three levels of the semantic network hierarchy are heuristically considered to be too general.

To get the semantic types, ST_A, of A-terms, we proceed as before but using the C-A relation filter. The ST_A list is then extended to include other semantic types that follow through an ISA relation with the semantic types in the ST_A list, if they are not too general. As an example, suppose "Pharmacologic Substance" is in the ST_A list. The semantic network indicates that there is an ISA relationship between this term and "Chemical Viewed Functionally." Therefore, the semantic type "Chemical Viewed Functionally" is added to the list ST_A. For each element semB in ST_B, if semB has at least one relationship in the B-A relation filter with any element in ST_A in the semantic network, we keep semB. Otherwise semB is dropped from the ST_B list. Next, the same is done to eliminate any semantic type in ST_A which has none of the relationships listed in the B-A relation filter with any element in the ST_B list. This step is known as *mutual qualification*. The final step in finding the semantic types of B- and A-terms is to remove, from the ST_B list, those semantic types that also appear in ST_A. At this stage, the semantic types of B- and A-terms are determined and are ready to be used by ExaminMED to prune out semantically irrelevant terms.

The pipes are then set properly in order to indicate which filters communicate with each other and with what data. In this case, as shown in Figure 3-1, all the four filters are connected through the pipes, since using semantic filtering was desired. Once the C-term is fed into the pipes, the coocBFilter submits queries to MEDLINE in order to find all

major MeSH headings (terms) that have co-occurred with the given C-term. This yields a list of potential B-terms, each of which is passed to semBFilter for further pruning. The semantic types of each B-term *b* are then retrieved by semBFilter through accessing the UMLSKS. If these semantic types are present in the ST_B list, then *b* is displayed to the user right away and is at the same time passed through the pipes to coocAFilter. Once coocAFilter receives a B-term, it queries MEDLINE to find potential A-terms, that is, all major MeSH headings that have co-occurred with that particular B-term but not with the initial C-term, which the user entered. The potential A-terms, once identified, are passed to semAFilter for further qualification. The semantic types of the potential A-terms are retrieved through UMLSKS, and if they appear in the ST_A list, the corresponding A-terms obtained are presented to the user. Figure 3-3 presents the above process steps in our proposed framework.

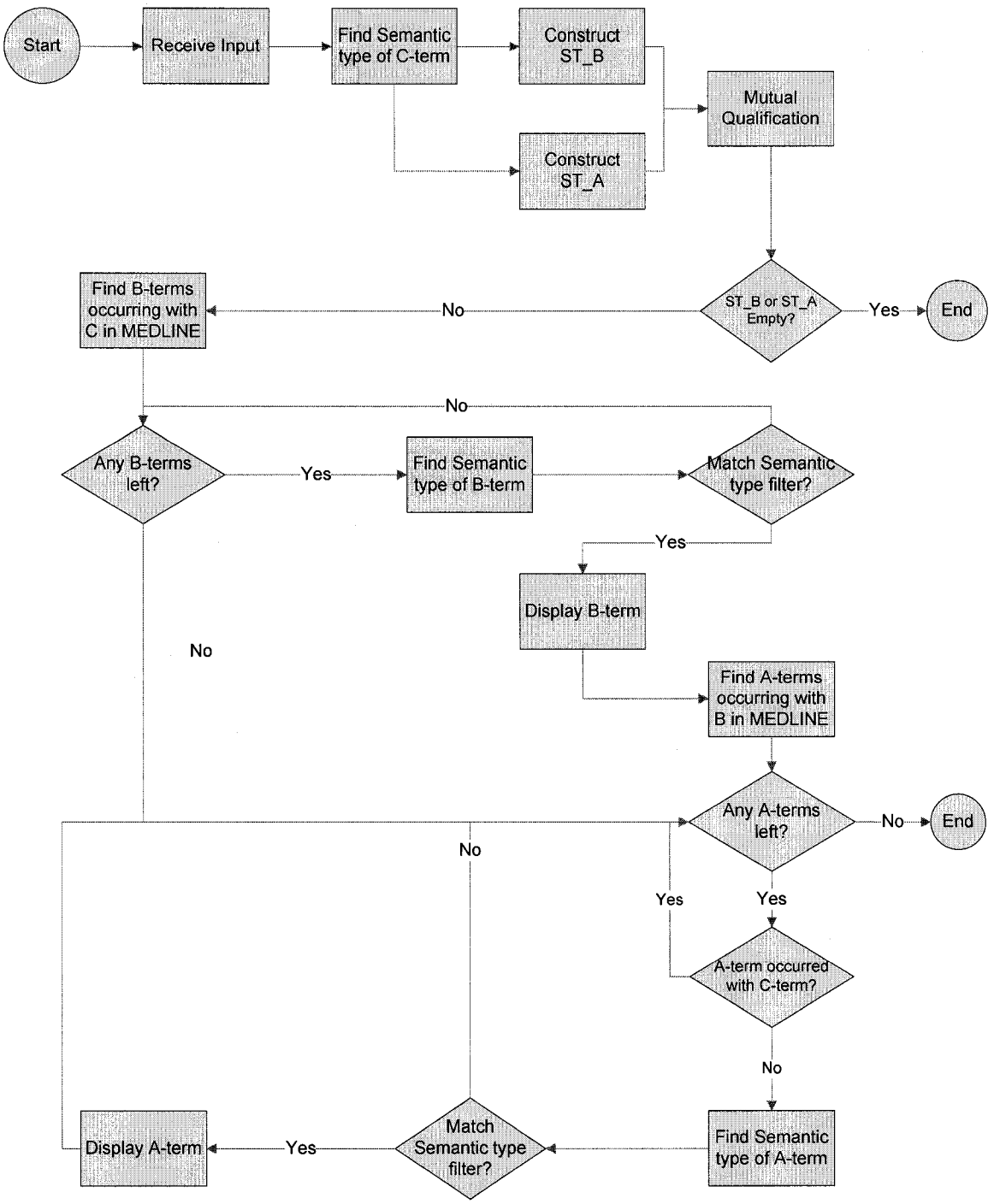


Figure 3-3. Process Flow Diagram using Semantic Filtering

If the user is not interested in pruning the terms based on their semantic types, then the only input required is the starting concept, the initial C-term. In this case, ExaminMED

connects to its local MEDLINE database and retrieves all the articles which contain the given C-term. All the major indexed MeSH headings in the retrieved articles are extracted and displayed to the user as B-terms. Each B-term *b* is then searched against MEDLINE, following the same steps as the initial C-term, and all major MeSH headings which co-occurred with *b* but not with the C-term are returned as A-terms. The process flow, when semantic filtering is not used, is presented in Figure 3-4.

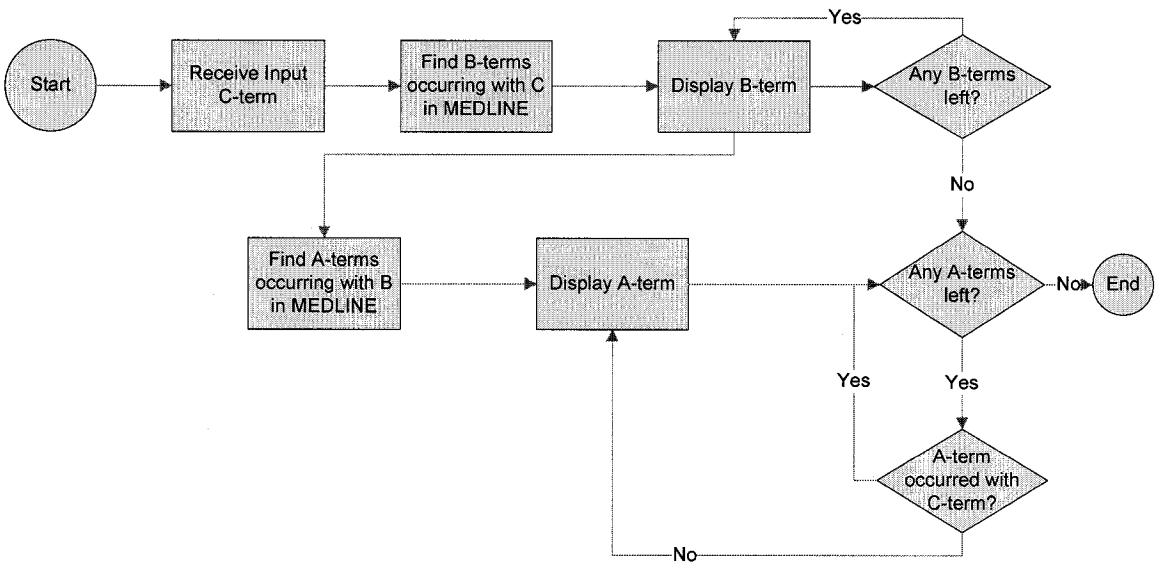


Figure 3-4. Process Flow Diagram not using Semantic Filtering

## *3.2 System Architecture*

In this section, we present the overall architecture of the system we developed. We also introduce its various modules and components and explain how they interact with each other.

Figure 3-5 shows the architecture of ExaminMED. As can be seen, it includes the following modules or components: MEDLINE database in XML format, BioText XML transformer, MEDLINE database in a relational database management system (RDBMS), UMLSKS server, the main engine or control module, and a simple graphical user interface (GUI).

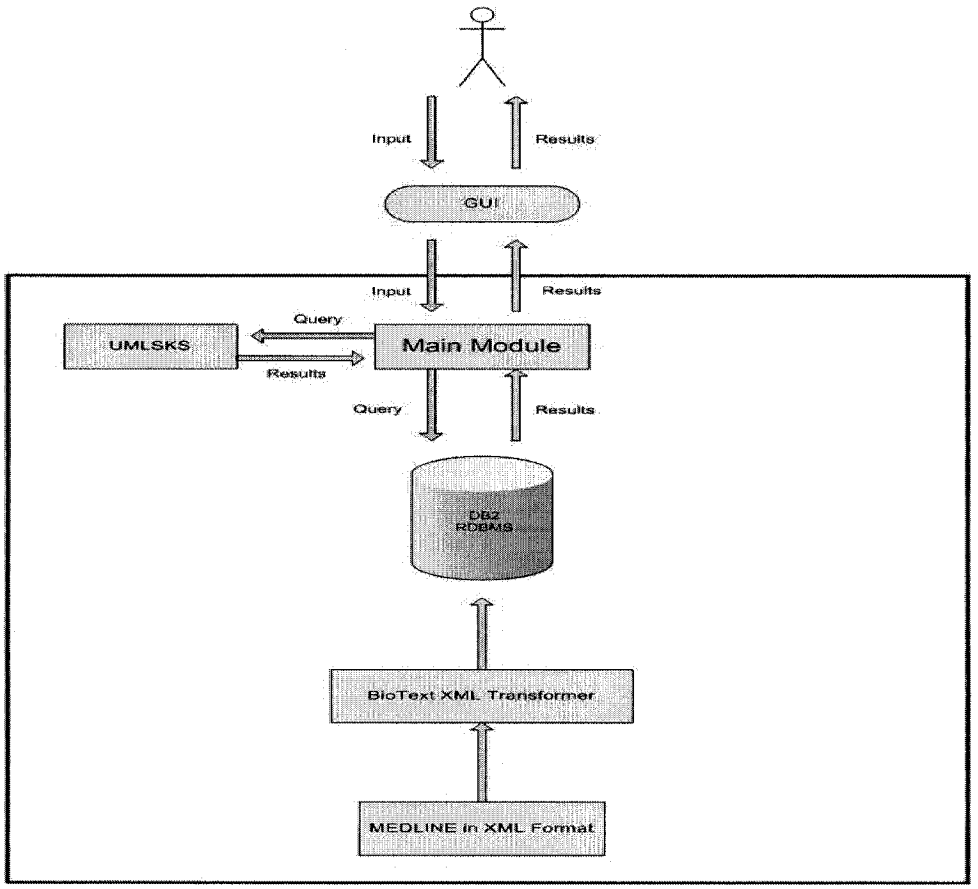


Figure 3-5. ExaminMED System Architecture

Since MEDLINE is available in XML format, it is first transformed into tables in our relational database developed using the IBM DB2 RDBMS, which supports efficient query processing and search operations needed in our work. We also use BioText, a tool developed at Berkeley, for transforming MEDLINE into these tables [3]. The GUI we

developed facilitates user's interaction with the system. Upon user's request, the system queries MEDLINE, while making use of the UMLS ontology, and returns the results to the user through the GUI, once they are ready. The main engine, which implements the processing steps and our approach explained above, connects to the UMLSKS, takes the input provided by the user, and poses queries to MEDLINE after communicating with the UMLS ontology to obtain synonym and semantic information. The results are returned to the user via the GUI. In what follows, we describe the components of the framework in detail.

### 3.2.1 Access and Transformation of MEDLINE

PubMed provides a web-based user interface and search engine which allows users to query the MEDLINE database online. However, it does not directly allow certain operations and queries that can be supported by RDBMS technology [15]. For example, in order to prevent server overload, PubMed limits the number of queries a user can perform in a given time interval. Also, queries whose results are large in volume should be performed at nights or on weekends. Storing a local copy of MEDLINE selectively in an RDBMS can give us greater control and does not have the limitations of querying MEDLIME through the PubMed interface.

As we did, users interested in querying a local copy of MEDLINE are required to obtain a license and provide NLM (National Library of Medicine) with contact information as well as information regarding the intent and kind of activities for using MEDLINE. Licensees may then download the MEDLINE database in XML format via FTP. In order

to be able to use RDBMS technology, we used the Java version of the BioText software to transform the database from XML format into a local relational database, for which we used IBM DB2. The main reason for using DB2 was to take advantage of RDBMS technology for storing, indexing, and querying large databases, such as MEDLINE. The powerful query processing and optimization techniques built in database systems make them attractive and useful especially when dealing with large databases. Another advantage of using RDBMS technology in our work is that it is usually easier to query a relational database than XML-formatted data. Relational databases are more standardized for structured data and support SQL (Structured Query Language), which is a "declarative" query language [15]. Finally, to formulate complex queries and combine search results in our work, we were interested in storing additional information such as the search results for various C-terms. All these requirements were accomplished in our work through using relational database technology.

.

### 3.2.2 Relational Database Management System (RDBMS)

The database schema used in BioText is made available in the Internet. The schema was designed using Document Type Definition (DTD) for MEDLINE, which defines the structure of the data in the XML files. The database schema includes 19 database tables, some of which have parent-child relationships. We have added two new tables to this database for specific needs in our application. Figure 3-6 shows these tables and their relationships. As the figure demonstrates, our database consists of 21 tables, 17 of which reference the main table, medline_citation, by referencing its primary key, PMID

(PubMed Identifier). Medline_mesh_heading_qualifier references the attributes PMID and descriptor_name in table medline_mesh_heading. The two new tables we added to the database are medline_sem_relations and medline_search_results.

Medline_sem_relations stores the semantic relations between the semantic types in the UMLS semantic network. UMLS provides this information in text format. We parsed the text file and populated the medline_sem_relations table accordingly. Our algorithm uses the information in this table to find the allowable semantic qualifiers of the B- and A-terms. This information could also be obtained through the UMLS API, which is explained in the next sub-section. However, for efficiency reason and to save processing time, we decided to store this information locally in our database and query it easily when needed. Since our prototype allows users to save and combine the search results, we also introduced the table medline_search_results. More explanation about the tables, their schemas and use follows.

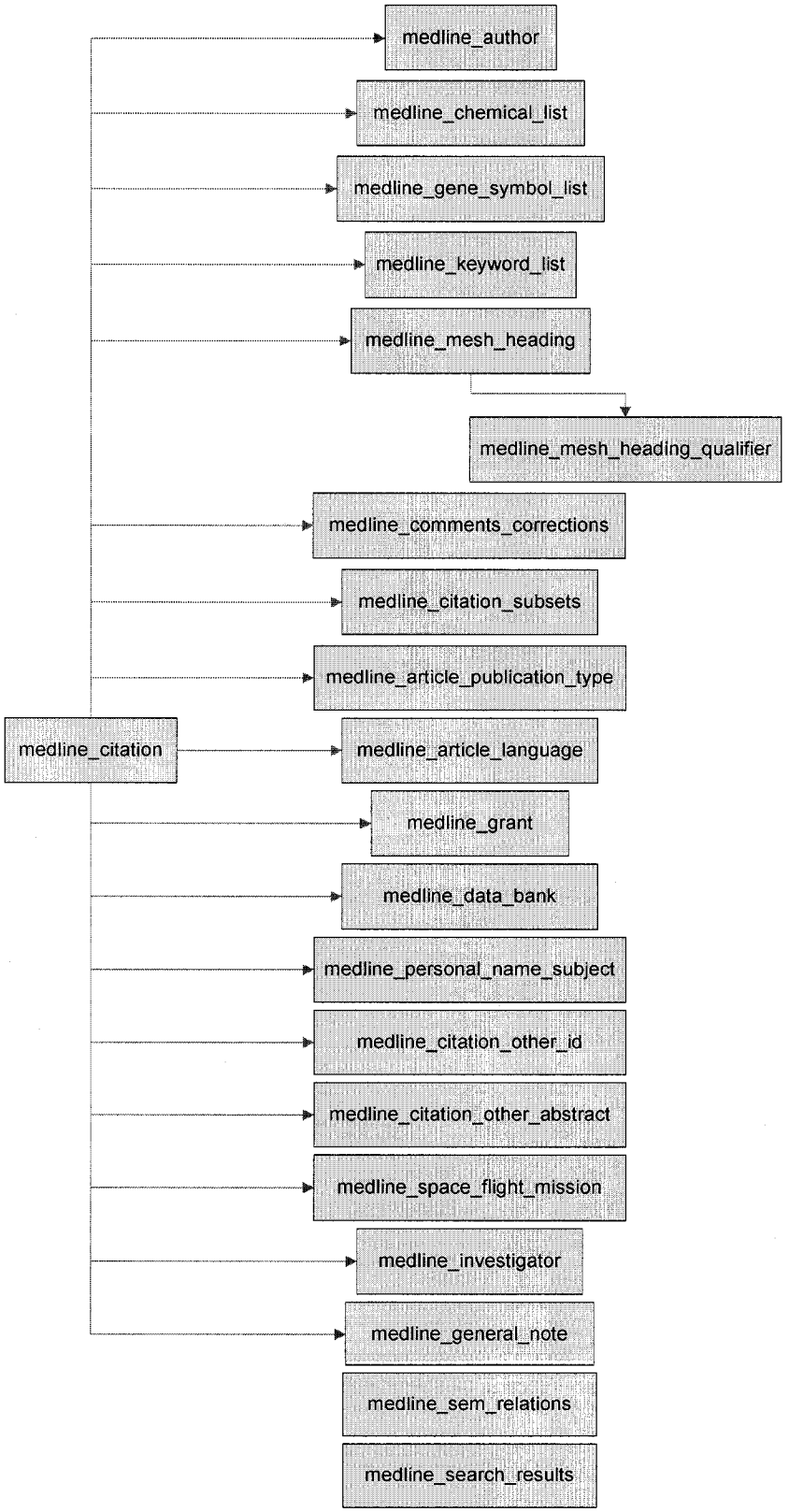


Figure 3-6. Database Table Names and Their Relationships

| Table Name | Table Attributes |
|---|---|
| **medline_citation** | **pmid**, date_created, date_completed, date_revised, issn, volume, issue, pub_date_year, pub_date_month, pub_date_day, pub_date_season, medline_date, journal_print_yn, coden, journal_title, iso_abbreviation, article_title, start_page, end_page, medline_pgn, abstract_text, copyright_info, article_affiliation, article_author_list_comp_yn, data_bank_list_comp_yn, grantlist_complete_yn, vernacular_title, date_of_electronic_publication, elec_pub_official_date_yn, country, medline_ta, nlm_unique_id, xml_file_name, number_of_references, keyword_list_owner, citation_owner, citation_status |
| **medline_author** | *pmid*, last_name, fore_name,first_name, middle_name, initials, suffix, affiliation, collective_name, dates_associated_with_name, name_qualifier, other_information, title_associated_with_name |
| **medline_chemical_list** | *pmid*, registry_number, name_of_substance |
| **medline_gene_symbol_list** | ***pmid*, gene_symbol** |
| **medline_keyword_list** | ***pmid*, keyword**, keyword_major_yn |
| **medline_mesh_heading** | ***pmid*,descriptor_name**,descriptor_name_major_yn |
| **medline_mesh_heading_qualifier** | ***pmid*, *descriptor_name*, qualifier_name**, qualifier_name_major_yn |
| **medline_comments_corrections** | *pmid*, ref_pmid, note, type, ref_source |
| **medline_citation_subsets** | ***pmid*, citation_subset** |
| **medline_article_publication_type** | *pmid*, publication_type |
| **medline_article_language** | *pmid*, language |
| **medline_grant** | ***pmid*, grant_id**, acronym, agency |
| **medline_data_bank** | *pmid*, data_bank_name, accession_number |
| **medline_personal_name_subject** | *pmid*, last_name, fore_name, first_name, middle_name, initials, suffix, dates_associated_with_name, name_qualifier, other_information, title_associated_with_name |
| **medline_citation_other_id** | *pmid*, source, other_id |
| **medline_citation_other_abstract** | *pmid*, type, copyright_info, abstract_text |
| **medline_space_flight_mission** | *pmid*, space_flight_mission |
| **medline_investigator** | *pmid*, last_name, fore_name, first_name, middle_name, initials, suffix, affiliation |
| **medline_general_note** | *pmid*, owner, general_note |
| **medline_sem_relations** | **left_side, relation, right_side** |
| **medline_search_results** | **cterm, bterm, aterm**, sem_filter, keep_term |

Table 3-1. Conceptual Database Design

The conceptual database design includes the relations (or tables) listed in Table 3-1. The primary key attributes are underlined and shown in **bold**, while foreign keys are shown in *italic*.

Our system prototype mainly queries the medline_mesh_heading, medline_sem_relations, and medline_search_results tables. Medline_mesh_heading contains information about the descriptors of MeSH headings in each entry in MEDLINE. This information is the result of indexing MEDLINE articles by NLM experts and indicates the content of the articles. We use the indexed MeSH terms to extract co-occurrence relations. Two MeSH terms are known to co-occur (with each other) if they are indexed in the same article as major headings.

Table medline_sem_relations contains the semantic relations between the various semantic types as indicated in the UMLS semantic network. This table is used to construct the ST_B and ST_A lists, which respectively are allowable semantic types for the B- and A-terms.

Finally, table medline_search_results stores the results of the user's interaction with the system upon request. A record is added to the table for each A-term retrieved. Attribute *cterm* represents the C-term in question. The second attribute, *bterm*, in this table contains the B-term that has occurred with the A-term in the aterm column. Attribute *sem_filter* records whether semantic filtering was used in deriving the A-term, and may have the values 'y' or 'n', denoting *yes* and *no,* respectively. The final attribute, *keep_term*, indicates whether the user chose to keep or discard a particular A-term while interacting with the system. The possible values for this attribute are 'y' and 'n', indicating keep and discard, respectively. For example, removing *Hypertension* from the

list of B-terms results in deleting the term *Adrenalectomy* from the list of A-terms. Thus, if semantic filtering was used and the user deleted *Hypertension* from the B-terms, then we create tuple (raynaud disease, hypertension, adrenalectomy, y, n) in table medline_search_results.

### 3.2.3 Unified Medical Language System Knowledge Source Server (UMLSKS)

The UMLSKS, provided by NLM, provides access to the three data sources in UMLS, namely the Metathesaurus, the Semantic Network, and the SPECIALIST Lexicon. UMLSKS can be accessed through the Internet, an XML-based socket programming interface, and an Application Programming Interface (API). Our system accesses the UMLSKS through the UMLS API, which is written entirely in Java. It provides developers with an extensive query facility and uses XML to request data from UMLS. It also provides Java developers with about 40 convenient functions to access detailed information contained in the three data sources [12].

The API that connects our system to UMLSK is based on Java's Remote Method Invocation (RMI) communications protocols. RMI allows methods of remote java objects to be invoked from other Java virtual machines. We also invoke in our system the methods provided in UMLSKS. For knowledge discovery process, ExaminMED uses the methods provided by the API to retrieve concept unique identifier (CUI), semantic types, and synonyms for C-, B-, and A-terms, which reside in the Metathesaurus. In UMLS, each CUI is associated with one or more semantic types. To find the semantic types of each term, we first need to retrieve its CUI through the API. We then pass the CUI to the

appropriate method in the API along with other necessary arguments. Finally, the method returns the semantic types in the form of an array.

### 3.2.4 The Main Module

This is the main module of our system which implements the algorithm and various search operations discussed above. It is also responsible for communicating with the other components. It receives user inputs, including C-term, via the GUI, queries the relational database for C, and finds relevant articles in which C has occurred as a major MeSH term. The major MeSH terms that appear in those articles are retrieved and their CUI are found. It also finds semantic types and synonyms through the UMLSKS, checks whether the semantic types match the ones in the ST_B list, and returns the B-terms through the GUI if they match the semantic type criteria. The search process continues in the same way in order to find relevant A-terms. Finally, the results are presented to the user through the GUI.

In the case where no semantic filter is selected, the main module can retrieve all the needed information directly from the RDBMS. It interacts with UMLS's Metathesaurus only when it is looking for synonyms of terms.

### 3.2.5 Simple Graphical User Interface

Our prototype system has a very simple and easy to use graphical user interface (GUI), which supports and facilitates user interactions with the system. The user enters the C-

term (s)he is interested in, accompanied by other inputs required by the system, including three relation type filters, and the role of the C-term. The resulting A-terms found are returned to the user in a way that is easy to read and comprehend.

It is important to keep in mind that the users of ExaminMED are medical specialists in general and not computer experts. They are often very busy and thus have limited time to spend on learning the software and how to use it. Also, they often face stringent time constraints thus requiring the explorations to be fast enough and easy enough to use. Therefore, it is important that the results be presented in a suitable and easy to understand manner, and in a reasonable amount of time. The goal is to allow medical specialists and researchers to focus more on hypothesis generation and discovering the unknowns rather than spend more time on learning to use the interface or manipulate the search results returned. This is why a very simple, easy to learn, and user-friendly interface is desired, and developed in our work. Figure 3-7 illustrates the GUI in our system. We elaborate on this interface and describe its functionalities in the next chapter.

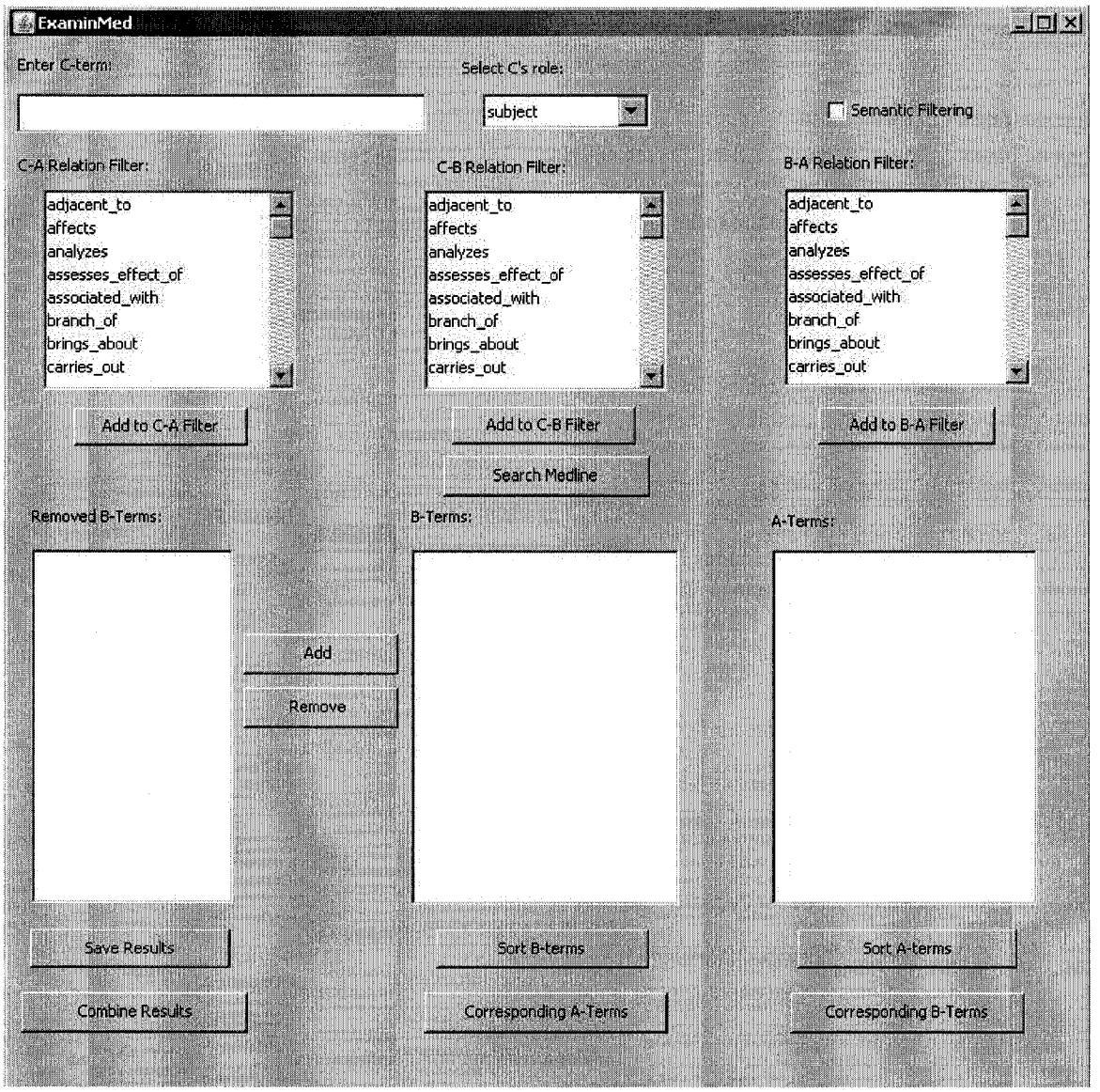


Figure 3-7. Graphical User Interface

# CHAPTER 4

# FEATURES AND ADVANTAGES OF ExaminMED

In the previous chapter, we proposed a framework, ExaminMED, which we implemented as a prototype for evaluation purposes. In this chapter, we present a prototype of the proposed framework, which is developed to illustrate the ideas, compare its features with existing systems, and demonstrate its advantages. The prototype developed implements a 'semi-automatic' solution to hypothesis generation and knowledge discovery in medical domain literature. We used records in MEDLINE as test data for our prototype. We used the idea behind Swanson's ABC model and proposed a framework which replicates an extension of this model, while containing various useful functionalities not present in existing systems. We used the pipes and filters architecture in the design of the system which makes it flexible and extensible, and allows concurrency through generation and execution of multiple threads, each of which is a filter. ExaminMED is interactive and allows user intervention. This exploratory interaction is an important characteristic of our system, as in our case the human intelligence is not completely automatable and replaceable by computers. We make use of the knowledge of the domain experts in order to limit the search to more meaningful B-terms and A-terms, and possibly generate valid hypotheses. Our simple and effective user interface is the key in supporting this *'human centered computation'* and receives inputs from the user and provides appropriate output in an incremental and easily 'digestible' fashion.

Our system differs from similar existing software tools in four major ways.

1. All the previous work which we reviewed directly used PubMed for querying MEDLINE. We have used a local copy of MEDLINE in an RDBMS to improve indexing and querying the data under the constraints of creating a local copy. This also eliminates some of the limitations that come with interfacing through PubMed.
2. Our framework is based on pipes and filters architecture which provides flexibility, extensibility, concurrency, the possibility of parallel execution, and other advantages that naturally come with incorporation of pipes and filters, which we will explain later in more detail. More importantly, incorporating this architecture allows us to create a framework, as opposed to typical software, which allows easy modification and improvement to the underlying system while assisting the UPK discovery task.
3. We also allow users to combine the results of different C-terms through using the Boolean operators *AND*, *OR*, and *NOT*. More complex filtering and combing results can be supported as SQL queries and evaluated using the database in our system. We elaborate this feature of our system in the next sub-section.
4. Medical researchers are the extremely busy people who have no time to learn either the database structure or a complex UI interface. They are often pressed for time and batch processing and in ordinate delays in iterative cycle could discourage them. The effective user interface that we have created is yet another positive aspect of our framework.

In the following section and sub-sections, we highlight the user interaction, computational, and non-functional features and advantages of ExaminMED.

## *4.1 User Interaction Features and Advantages*

Figure 3-7 showed a screenshot of the graphical user interface in our system prototype. It consists of sub-windows some of which have slide bars, buttons, combo box, and a drop-down list for selection purposes. Once the stage by stage progression of the ABC model, i.e., progressing from the input C-terms to the target A-terms, is understood, the UI is very intuitive to work with. The user interface provides various features and advantages, such as flexibility in choosing filters, freedom of refining results, saving and combining search results, demonstrating corresponding terms, and sorting results based on specific criteria. To demonstrate the features in action, we submit the following input to ExaminMED, the output of which is shown in Figure 4-1:

C-term: *Raynaud Disease*

Role: *object*

Semantic type filtering: selected

C-A relation filter: *treats, prevents*

C-B relation filter: *process_of, result_of, manifestation_of, causes, affects*

B-A relation filter: *interacts_with, produces, complicates, affects*

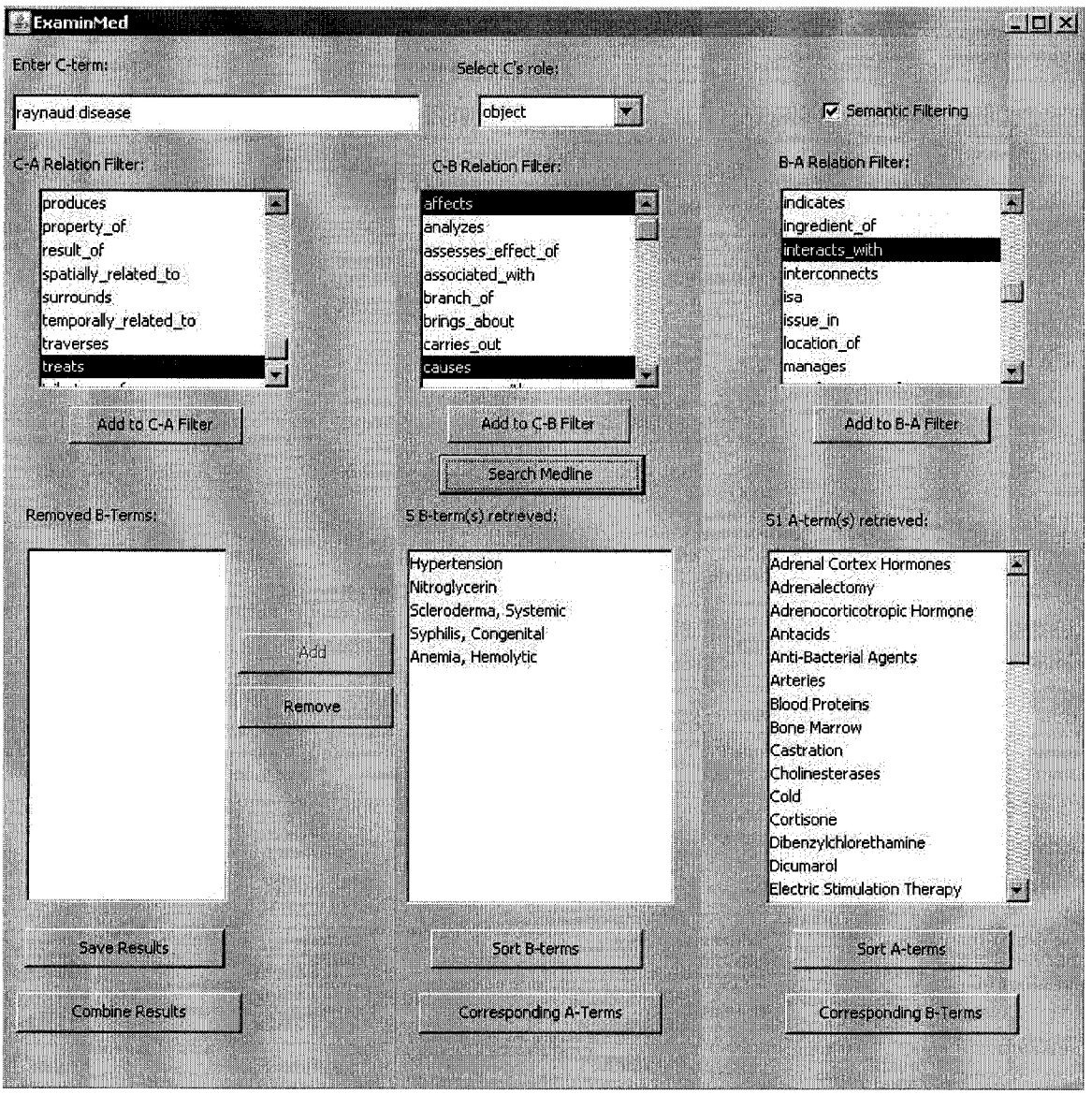


Figure 4-1. Results Displayed for the Raynaud Disease Example

### 4.1.1 Flexibility in Selecting Desired Filters

The GUI provides a check box for each of the filtering mechanism that ExaminMED provides. The user may select the desired filters which are to be used for pruning the B-terms and A-terms. ExaminMED then arranges the pipes so that only the desired filters

are connected and used in the pruning process. This flexibility at the user level allows the user to have more control over the system and provides a means to experiment with the results of different combinations of filters and perhaps obtain more meaningful and refined results. For example, the user can choose not to use the semantic type restrictions or wish to further explore the results of a search with or without the semantic type filters. In other words, the user has indirect control over the pipelines and how they are set and arranged.

Currently, ExaminMED provides two types of filters, namely co-occurrence filters and semantic filters. It is necessary for the co-occurrence filters to be present in all searches since the relationship between two terms is established through the co-occurrence of the terms in the literature. In other words, co-occurrence is the relationship that links the A-B and B-C terms. The involvement of the semantic filters, however, is not mandatory, but optional, in the UPK discovery process and therefore the ExaminMED gives the user the option to eliminate semantic filtering as desired.

For example, a user may want to view the B-terms and A-terms without considering their semantic types and strictly base the search on co-occurrence relations. In this case, (s)he can choose to eliminate the semantic type filter and investigate a larger number of terms, which have not been pruned by the semantic filters. In Figure 4-2, the results are shown in the case where the user chose not to use semantic filtering. Clearly, the number of remaining B-terms and A-terms is larger in this case since no semantic filter is applied (13 B-terms and 563 A-terms without semantic filtering versus 5 B-terms and 51 A-terms with semantic filtering).

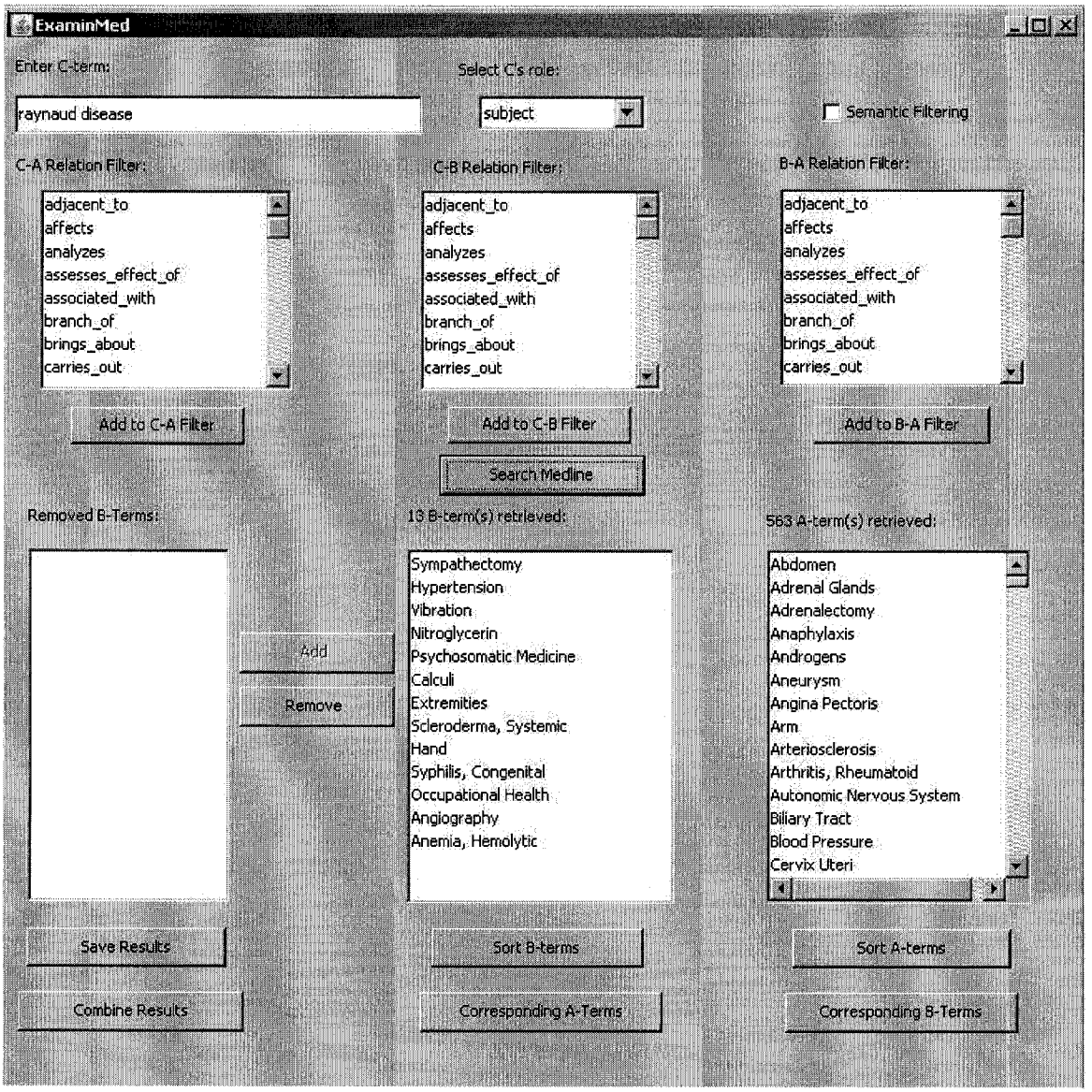


Figure 4-2. Semantic Filter not Selected

### 4.1.2 Refining Returned Results

As mentioned earlier, our system allows human intervention and interaction to take advantage of the expertise of the user and also to allow exploring the results for generating possibly new valid hypotheses. Since UPK discovery is an exploratory task,

ExaminMED provides the option of refining the terms by adding/removing them from the list. The user may want to eliminate one or more B-terms perhaps because they are not interesting or related to what (s)he has in mind. If the user is certain that some particular B-terms are not of interest, removing them may significantly reduce the number of A-terms which may be further explored by the user. When a B-term *b* is removed, all the A-terms that co-occurred with *b* are removed unless the A-term *a* had also occurred with other B-terms, in which case *a* is not removed. The removed B-term, *b* is moved to the leftmost list in the interface and the label above the rightmost list demonstrates the total number of A-terms that have been removed. For example, consider Figure 4-1 which showed the case where the C-term is *Raynaud Disease*. If the user decided that *Hypertension* is not as interesting B-term, (s)he can remove it by simply selecting this term in the list and then clicking the "Remove" button. Originally, *Hypertension* co-occurred with 177 A-terms in our local database. After applying the semantic filter, 39 A-terms remain in the A-term list. However, not all these terms are removed when the user eliminates *Hypertension*. *Cortisone*, for example, occurs with *"Hypertension"*, *"Scleroderma, Systemic"*, and *"Anemia, Hemolytic,"* hence it continues to remain on the list after removing *Hypertension*. Figure 4-3 shows the GUI when *Hypertension* is removed from the list together with all the 36 A-terms that appeared only with this term.

The removed B-terms are displayed on the leftmost list in the GUI. These B-terms and their corresponding A-terms can be returned to the original lists by selecting the terms and clicking the "Add" button. This is useful in case the user wants to perform "if-then" types of queries to investigate co-occurrence relationships in the results when some B-

terms are removed. In this case it is desired to get back the terms that are temporarily removed.

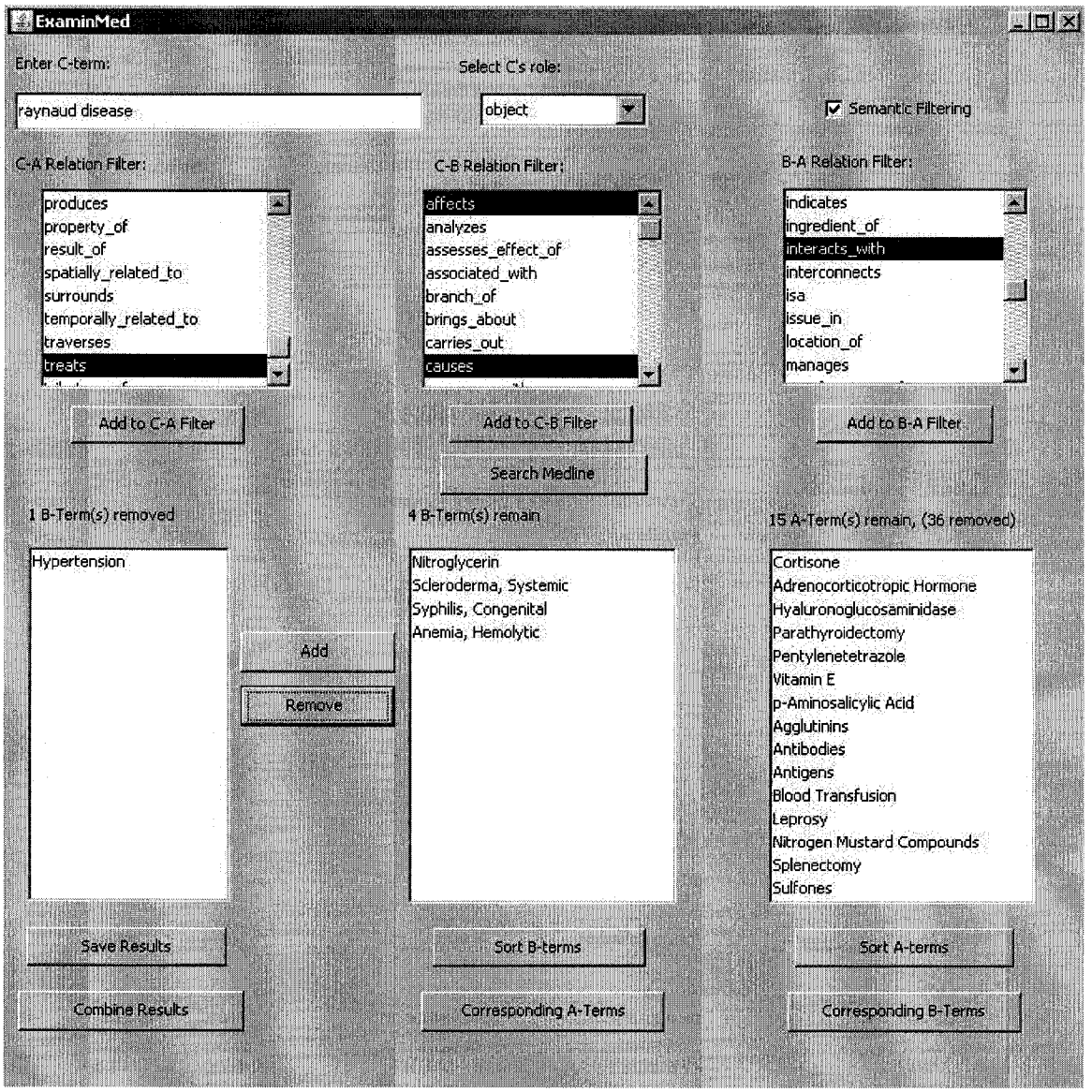


Figure 4-3. Removing the B-term "Hypertension"

### 4.1.3 Saving and Combining Search Results using SQL

The user may save the results in text files and also in the database for future reference. The saved text files reflect the latest interactions of the user, which includes the C-, B-, and A-terms, as well as the terms that were removed by the user and whether or not semantic filtering was performed in that particular search. This information is also saved in the database. The saved information can be used later for combining the results of various C-term searches through *AND*, *OR*, and *NOT* operations. Other functions and operations can also be added to the framework in the future. Presently, the logical operations are incorporated as examples to illustrate the idea. Figure 4-4 illustrates the GUI that appears upon requesting to combine the results of various searches.

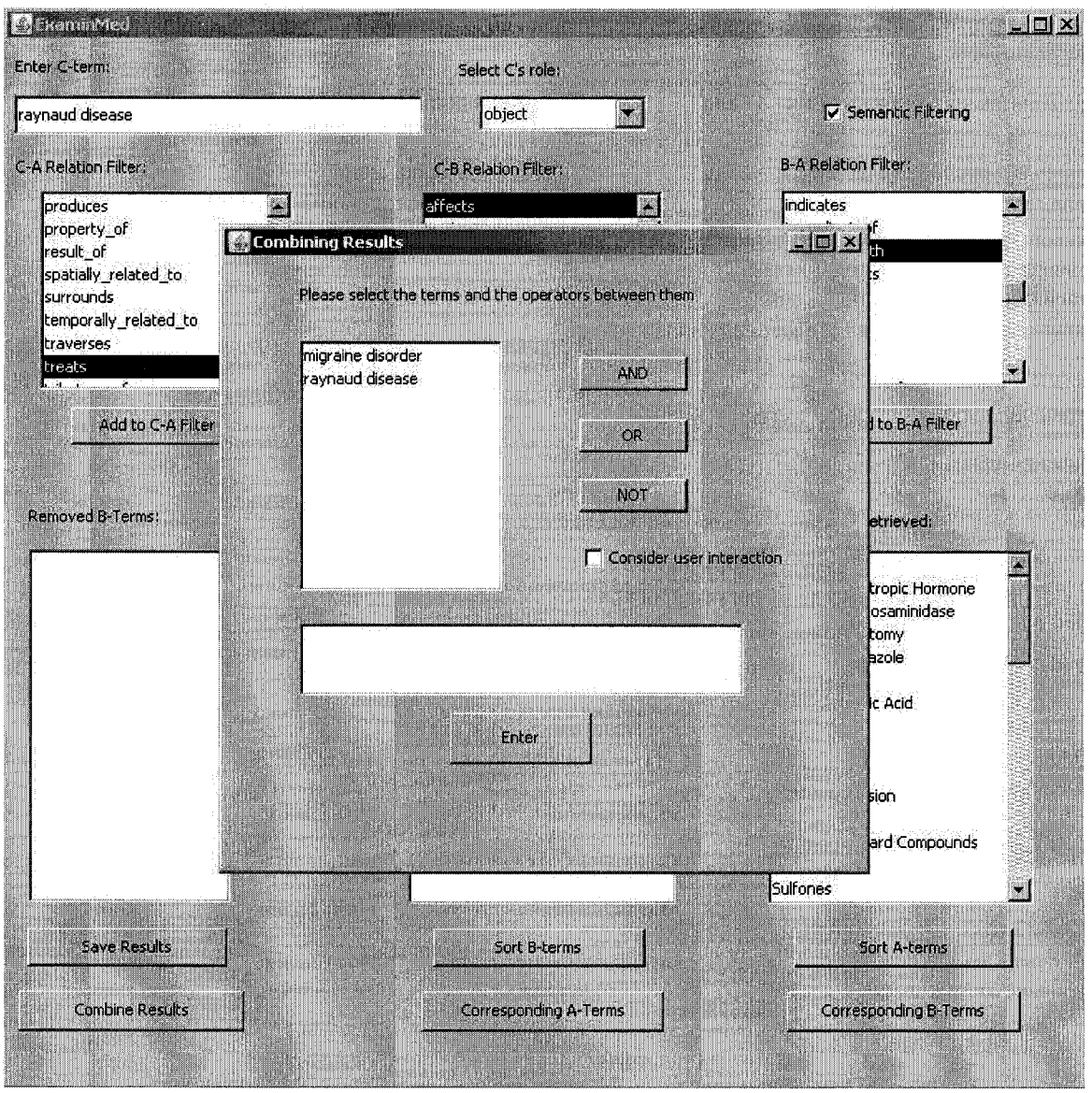


Figure 4-4. Combining Results Window

The user can click on a C-term as the first operand, and then choose a desired logical operator from the list, followed by the second operand chosen from the list. The text field at the bottom of the window displays the corresponding relational query. For example, the user may click on "Migraine Disorder" as the first term, then click the *AND* button, and then choose "Raynaud Disease" as the second term. In this case, the text field

displays the following text: "Migraine Disorder AND Raynaud Disease," as demonstrated in Figure 4-5. By clicking the "Enter" button, a new window appears which shows a list of all A-terms that the two individual searches of Migraine Disorder" and "Raynaud Disease" had in common. If the operator used is *OR,* instead of *AND*, then all the A-terms associated with "Migraine Disorder" would be displayed as well as those associated with "Raynaud Disease."

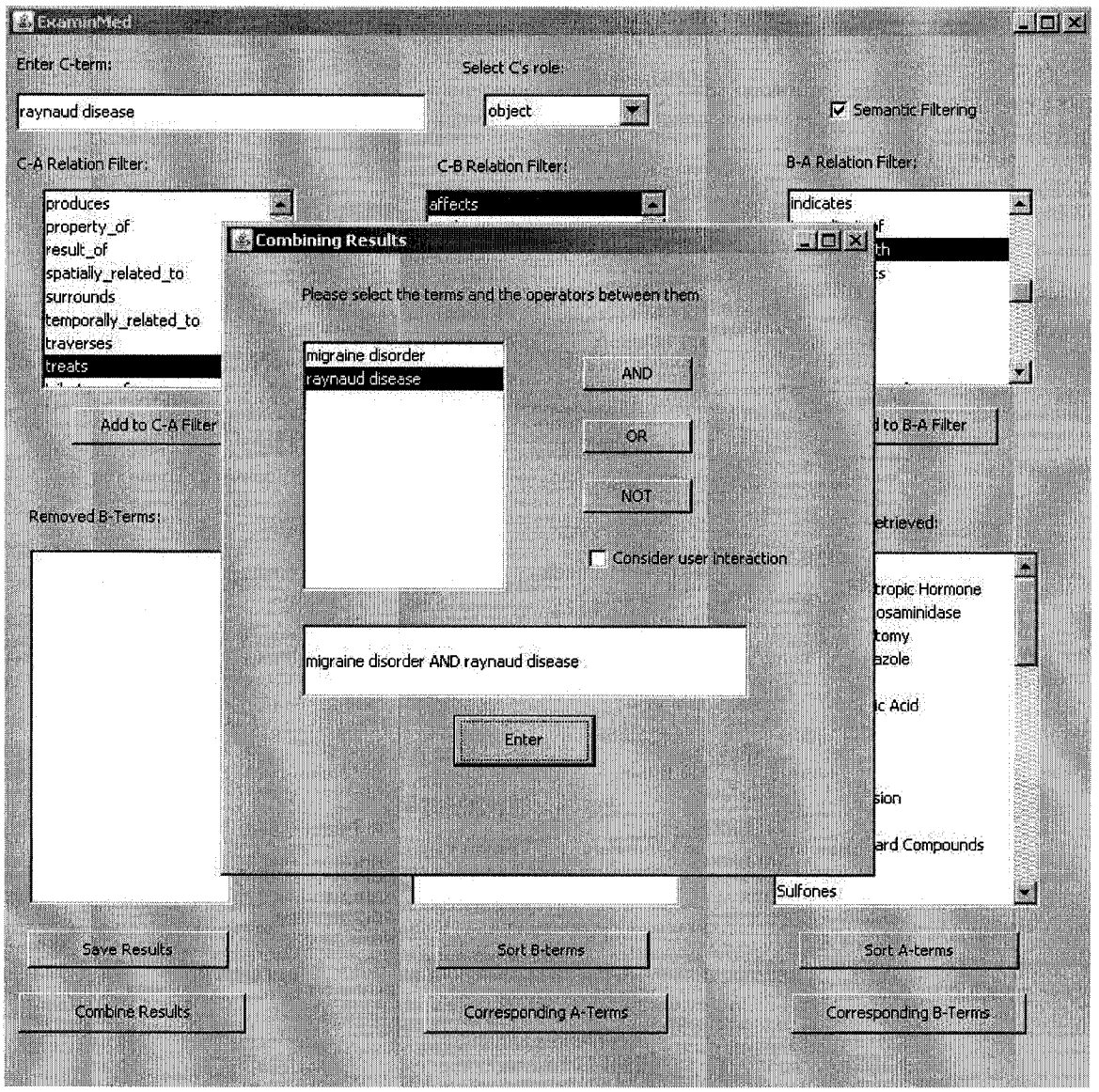


Figure 4-5. Combining Results Query

Before requesting the system to process the query, the user can choose whether or not the system should take his/her interactions into consideration by selecting/deselecting the "Consider user interaction" checkbox. If the checkbox is selected, then the terms which were eliminated by the user will not contribute to the results. Also if, for example, semantic filtering was intended during the search, only those A-terms that matched the semantic type criteria would be presented to the user. In Figure 4-6, we demonstrate the result of the query: "Migraine Disorder AND Raynaud Disease." In this case, the user's interactions with the system were not considered. Thus, the 92 A-terms which are displayed are those which these two C-terms had in common without using any semantic filtering.

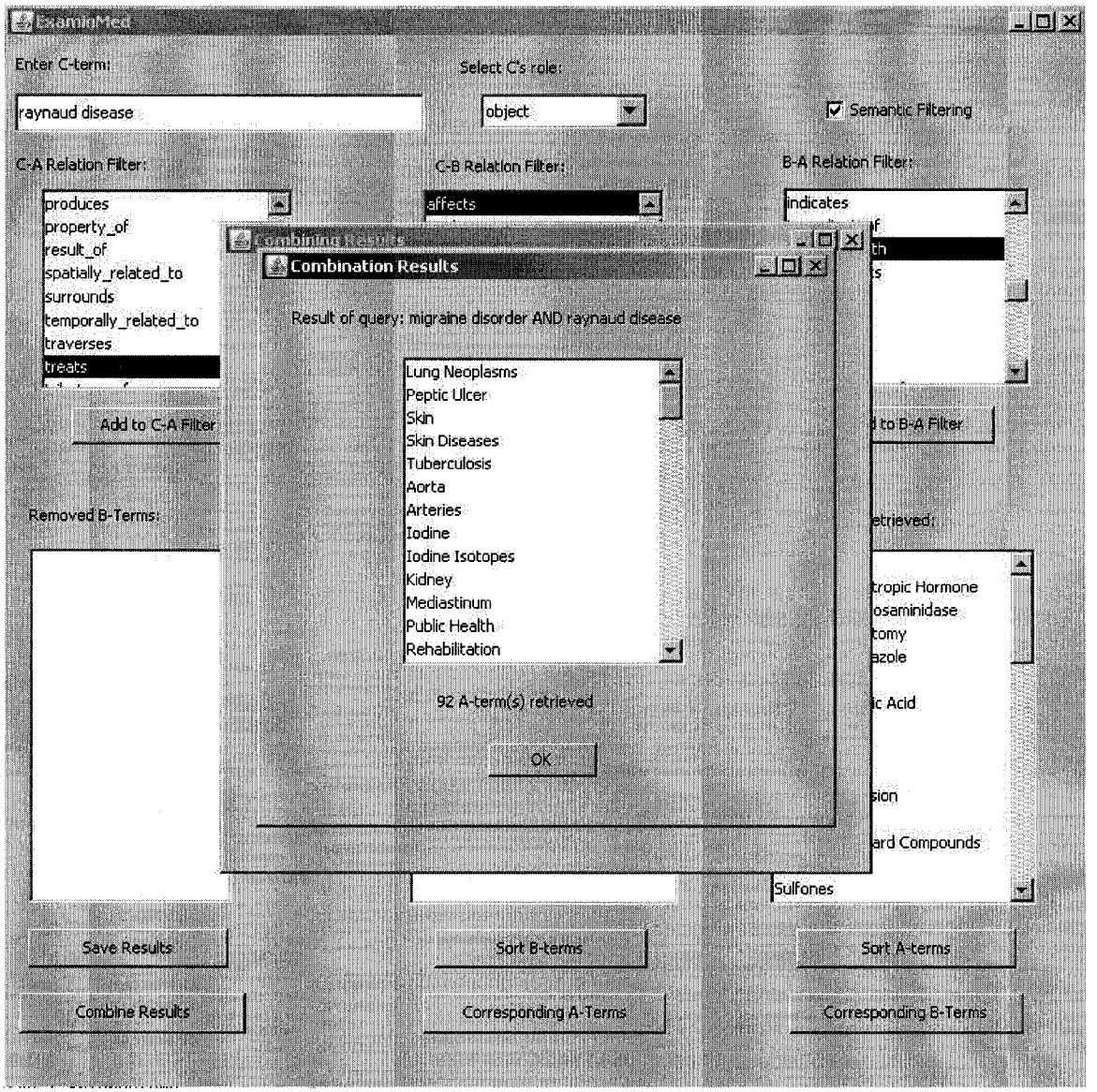


Figure 4-6. Results of Query: "Migraine Disorder and Raynaud Disease"

By allowing the user to combine the results obtained by searching various C-terms, our system provides more complex filtering. In a way, we are providing multiple pipes whose results merge in the last step depending on the user's request. For example, suppose that the user wishes to combine the results of two earlier searches conducted and saved using the two C-terms $C_1$ and $C_2$. That is, the user is interested to evaluate $C_1$ AND $C_2$,

assuming that the semantic filtering used in the two searches was desired. The result of this combined query can be essentially expressed by setting up two pipes as shown in Figure 4-7.

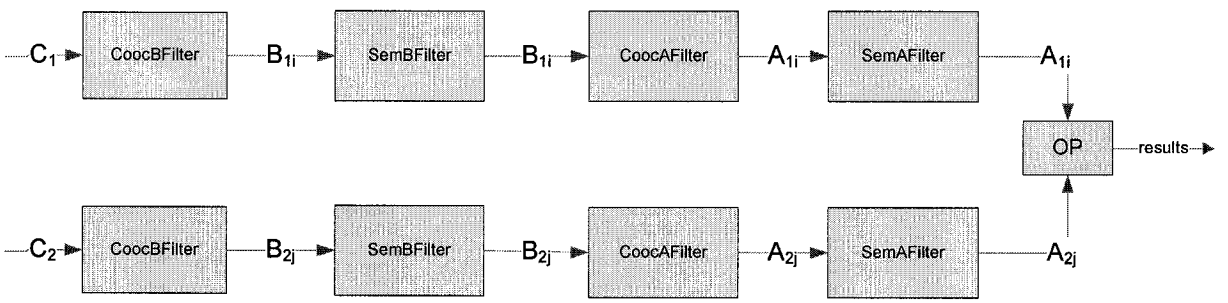


Figure 4-7. Pipe Setup for Query "C1 AND C2"

In general, the filters that the C-terms are fed into need not be the same. Each C-term may go through various different filters and the resulting A-terms can be combined through the desired operation. Also, the combining operator OP shown on the right in this figure could be *AND*, *OR*, and *NOT,* as well as more complex functions and combinations. In addition, a complex combination can involve many C-terms and operations as desired. Figure 4-8 illustrates this point. In this figure, each C-term is fed into appropriate pipes and its results are combined as defined by the user.

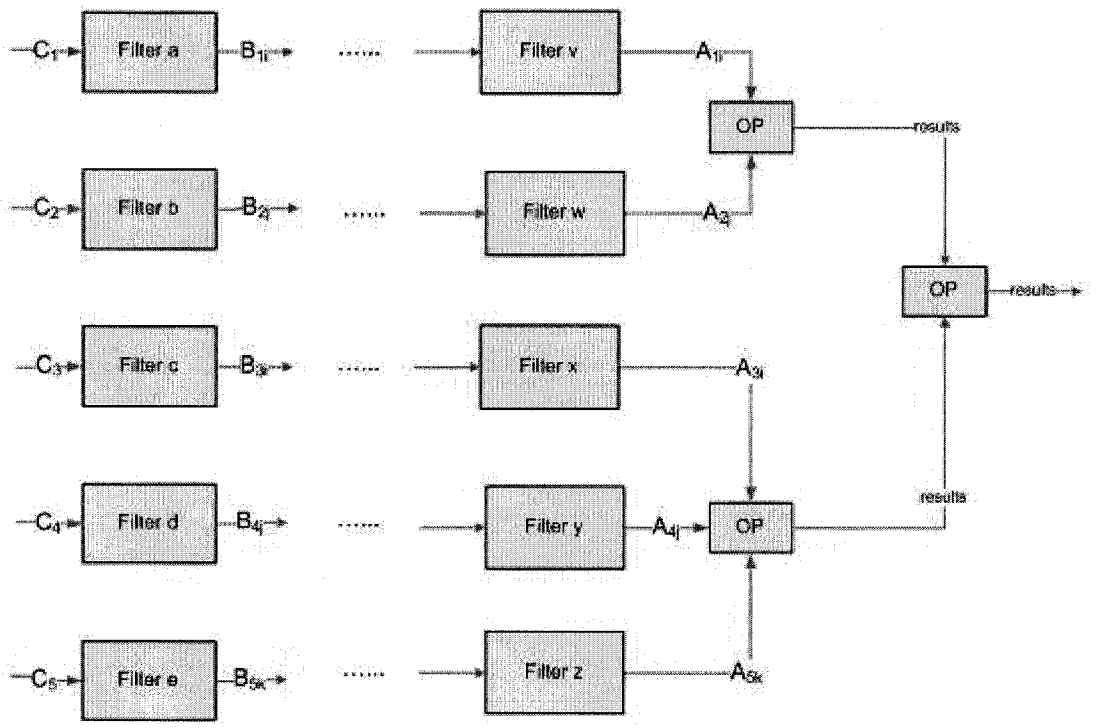


Figure 4-8. Pipe Setup for Complex Combinations

### 4.1.4 Viewing Corresponding Terms

It may be interesting for the user to know which B-terms and A-terms correspond with each other and get a feel for which B-terms and A-terms appeared together in MEDLINE for analysis purposes. The user can see which A-terms occurred with which B-terms, and vice versa by either placing the curser on that particular term or by selecting the term and clicking on the "Corresponding A-term" or "Corresponding B-term" buttons. In the first case, a tooltip appears which contains the list of all A-terms (B-terms) that have co-occurred with that B-term (A-term). In the second case, a new window pops up which

displays the list of co-occurring terms. Figure 4-9 illustrates an example, showing the B-terms that have appeared with the A-term *Cortisone*.

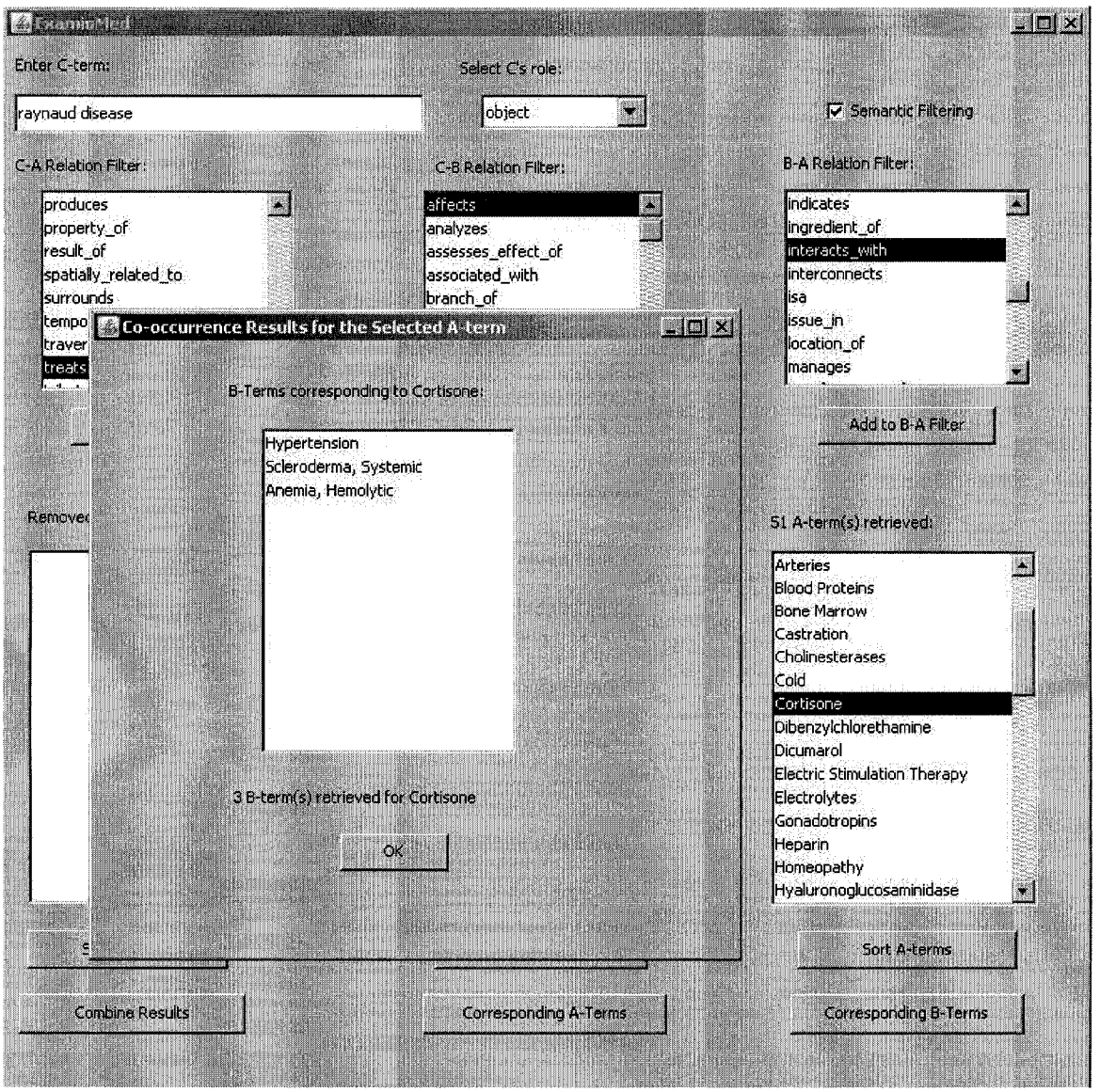


Figure 4-9. B-terms Corresponding to Cortisone

### 4.1.5 Sorting Terms

As the ExaminMED screenshots illustrate, the GUI contains two lists which display the B-terms and A-terms that have passed the filters. The lists are updated each time a new term is retrieved. The user is notified once the whole database is searched and the result is provided to the user through the GUI. Once the search process is complete, the user may sort the results which appear on the list.

ExaminMED keeps track of the number of articles in which each term appears. The "Sort B-Terms" button sorts the B-terms based on the number of articles in which they appeared with C-term. We suspect that a higher ranked B-term is perhaps more strongly related to the C-term in question. The "Sort A-Terms" button, on the other hand, sorts the A-terms based on the number of B-terms they have co-occurred with. We have chosen this sorting scheme because perhaps A-terms with a higher number of B-terms linking them to the input C-term indicate a stronger relationship with the C-term and are more strongly connected to it. Figure 4-10 shows the interface where the A-terms are listed. As the figure illustrates, *Cortisone* is rated as highest because it co-occurred with three B-terms in the local database. The second top-ranked term shown is *Adrenocorticotropic Hormone*, which has co-occurred with two B-terms, namely *Hypertension* and *Anemia, Hemolytic*.

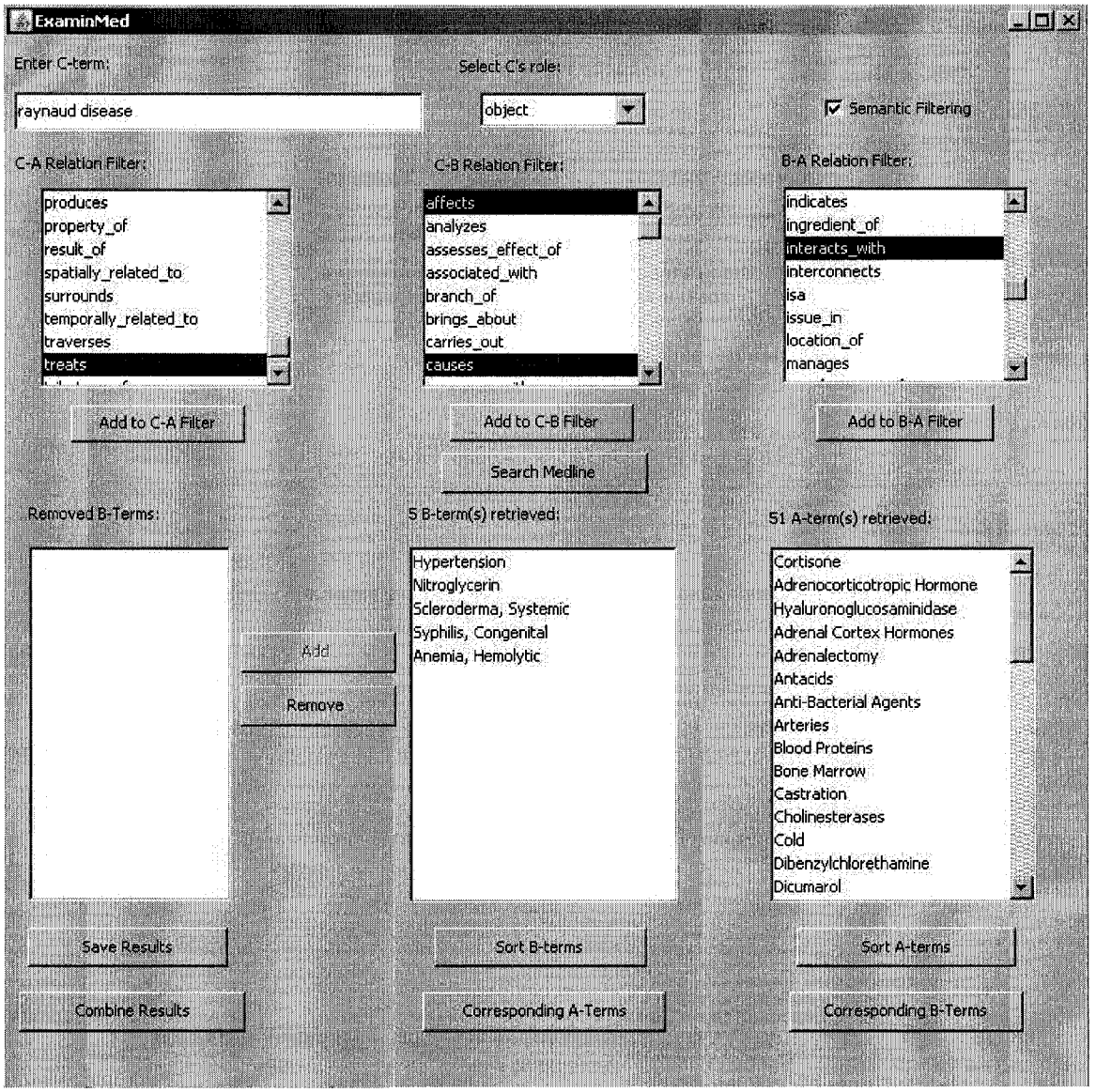


Figure 4-10. Sorted A-terms

## *4.2 Computational Features and Advantages*

In this section we discuss various computational features and advantages provided by ExaminMED. These include flexibility at the user level as well as architectural level,

efficiency, which is achieved through concurrent execution of various stages of the process, and use of RDBMS technology for storing and accessing MEDLINE.

### 4.2.1 Flexibility Due to Pipes and Filters

Our framework provides two levels of flexibility which can be taken advantage of, namely flexibility at the user interface level, which was discussed in 4.1.1, and flexibility at the architectural level. By using the pipes and filters architecture, both kinds of flexibility can be achieved.

We recall that at the user interface level, we allow the users to choose among the various pruning mechanisms and thus provide a flexible system with more freedom for users while conducting their explorations for potential hypothesis generations. Flexibility is an important quality that a UPK discovery system should support, since the task is an exploratory one and the more flexibility we provide, the more 'user-centered' the system is. Flexibility allows more freedom to the user for exploring and experimenting with various search options.

At the architecture level, our system can be easily enhanced by replacing existing filters with new ones or combining already available filters. The use of pipes and filters architecture also allows rapid prototyping and maintainability, since filters can be easily recombined, replaced, and reused. Our framework can benefit from these advantages to provide the possibility for enhancement and improvement.

### 4.2.2 Concurrency Support

Concurrency is another additional benefit provided by pipes and filters. This architecture allows concurrent execution of the filtering steps as each step is performed by a filter which is an independent component and runs as a separate process or thread. Hence, multi-threading, and as a result efficiency, are other benefits of using this architecture.

Since the filters in ExaminMED can operate independent of each other, they can execute concurrently as separate threads as long as their required input is available. In our framework, filtering occurs concurrently when computing B-terms and their corresponding A-terms, in a pipeline fashion. This allows early interaction of the user while the system is busy computing more terms. This incremental response is useful upon noting the size of the database, and thus the large number of terms that needs to be processed and returned to the user as output. It is more convenient for the user to view each result right away as it is produced, rather than being bombarded with the complete and possibly large set of terms at once after the search process terminates. This feature allows the user to reflect on a smaller number of terms while more results are being produced by the system.

### 4.2.3 Use of Database Technology to Store and Access MEDLINE

As noted previously, existing hypothesis generation systems query MEDLINE through PubMed's web-based interface and search engine. While useful, using PubMed poses certain limitations, such as server overload. For instance, users may enter a limited

number of queries within a given time interval, queries which produce large results cannot be performed other than during the nights and weekends to avoid server overload. Storing MEDLINE in an RDBMS in our work allows greater control and higher query ranges supported by this technology. Database technology supports management of large data and efficient query processing and optimization. A wide variety of queries can be performed in relational database systems, some of which are not possible or not easy to perform through PubMed [15]. Queries that are hard to formulate and evaluate in PubMed can be easily expressed in SQL. For example, MEDLINE can be queried to rank journals based on the number of articles in those journals which contain a certain MeSH term. Also, DB2 allows indexing of text in the database by using the keyword *CONTAINS*, which is not part of the standard SQL language.

## *4.3 Non-Functional Features and Advantages*

This section presents the non-functional features and advantages which ExaminMED provides. The following sub-sections discuss extensibility, usability, and scalability issues addressed by ExaminMED.

### 4.3.1 Extensibility

One of our main goals in this thesis was to build a framework which can be easily extended. We realized the need for such framework after learning about the various attempts made by researchers to extend the basic ABC model proposed by Swanson. The

pipes and filters architecture provides extensibility, since new filters can be easily added to the system and connected to other filters through the pipes in a rather straightforward manner. The possibility of easy addition of filters is due to decoupling, which is the derived advantage of using the pipes and filters architecture. Decoupling is the result of having independent filters which do not communicate with or depend on each other except through the data stream in the pipes, as is the case for our application. Therefore, adding new filters does not affect the rest of the system and only requires setting the pipes properly to connect the new filters to other existing ones. This quality makes pipes and filters a suitable architecture for creating a software framework designed for applications such as ours. Also, the framework can be extended to operate on databases other than MEDLINE. As long as the criteria for relating terms in the database are defined, ExaminMED can apply the ABC model for UPK discovery.

### 4.3.2 Usability

The discovery of UPK is an interactive and 'user-centered' task. Therefore, it is crucial to consider user needs and restrictions when designing and developing software that is intended to be interactive. So far, all systems that implement the ABC model for hypothesis generation have either relied too heavily on the user or have completely eliminated human interaction. ExaminMED provides a trade-off between the two options. We allow the user to interact with the system if (s)he chooses to do so. If the user decides not to interact with the system, the system can still perform the computation and yield meaningful terms.

ExaminMED is a user-centered software framework which allows the user to easily explore and experiment with the results for the hypothesis generation task. As mentioned before, our prototype provides the possibility of choosing desired filters, refining results, saving and combining search results, sorting results, and viewing corresponding terms, all of which allow the user to be actively involved in the UPK discovery process. We elaborate more on this feature in the following chapter where we discuss the usefulness of ExaminMED.

### 4.3.3 Scalability

Due to the architectural basis of ExaminMED, concurrent execution of multiple threads, each of which representing a filter, is possible since the filters are decoupled and do not depend on one another. In addition to concurrency advantage, concurrent computation can be realized through pipeline processing or parallelism, when multiple resources including CPUs are available, resulting in increased performance. With multiple CPUs, ExaminMED can scale to larger databases, such as the entire MEDLINE. For example, one CPU may execute the first filter, while others execute the next filters once the required input is available to them. We remark that while we rely on database technology to store and search large amount of data efficiently, the performance of the proposed system when processing the B-terms and A-terms to identify co-occurrences is not dependent on the size of the database, but on the number of these co-occurrences and associations between the terms.

# CHAPTER 5

# IMPLEMENTATION AND EVALUATION

In the previous chapters we proposed a flexible, extensible, concurrent, and interactive framework based on Swanson's ABC model, for which we also proposed using pipes and filters as the architectural design and discussed the advantages of our approach. We have developed a running prototype and performed experiments to illustrate our ideas and evaluate various aspects of the system. This chapter reports our implementation details and experimental results.

## *5.1 Implementation Details*

Our prototype system is developed using the JBUILDER 2006 IDE, and implemented in Java version 1.6.0, with over 2700 lines of code. We use the IBM DB2 version 1.9.0 as the database management system to store, query, and manage our local MEDLINE database. Our prototype runs on a conventional desktop, with Intel Pentium 4 CPU at 2.80 GHz, 1GB RAM, single-threaded, and running Windows XP.

Since our main objective in implementing a prototype system was to evaluate the proposed ideas for an experimental investigation, we only store and use a relatively small portion of MEDLINE, as opposed to the entire database. We made the decision to use a subset of the database mainly because of the fact that the large amount of time required to transform all the entries in MEDLINE from XML format into the RDBMS is not available within the scope of a master's thesis prototyping. As a reminder, the MEDLINE

database is distributed in over 500 XML files, each of which takes several minutes to transfer to relational tables in DB2. Demonstrating that the prototype produces meaningful results with a portion of MEDLINE is considered enough to show that it can be scaled up with proper techniques. Our database consists of 76,839 entries, each representing a MEDLINE article; the whole MEDLINE includes about 16 million entries. The entries in our database are extracted from the first 6 MEDLINE XML files. From our understanding, the MEDLINE XML files and their contents are not organized according to any particular criteria, and hence in a way the articles in our database are not chosen on any specific basis to avoid biased results.

Our prototype consists of 11 java files, 10 of which contain a single class each. They are introduced in Table 5-1. Of these, 8 classes extend Java's Frame class and are developed and used as interfaces to display different information to the user upon request. For instance, one of these frames, *coocResults*, displays the corresponding B-terms for the A-term selected by the user. The modules *ConnectToDatabase*, *DBFacade*, and *Main* are the only java classes in our implementation that do not have an inheritance relationship with other java library classes.

The main class is *MainFrame* which basically implements the core algorithm, and contains 6 other classes, namely *mainThread*, *producer*, *coocBFilter*, *coocAFilter*, *semBFilter*, and *semAFitler*, each of which extends Java's Thread class and runs as a separate thread. Class *mainThread* is responsible for connecting, through the pipes, the various filters chosen by the user. If the user selects the option of including semantic filtering in the pruning process, then *mainThread* connects the filters *producer*, *coocBFilter*, *coocAFilter*, *semBFilter*, and *semAFitler* through the pipes. On the other

hand, if the user has not selected semantic pruning, the only filters participating in the pruning would be instances of *producer*, *coocBFilter*, and *coocAFilter*.

| Class | Functionality |
|---|---|
| ChooseSemType | Pops up if C-term has multiple semantic types associated with it. It allows the user to choose a desired semantic type. |
| CombinationResult | Shows the A-terms obtained by combining the results of various C-terms using AND, OR, and NOT operators. |
| Combine | Pops up when the "Combine Results" button is clicked. It displays the C-terms available in the database and allows the user to express queries for combining the results. |
| ConnectToDatabase | This class creates the connection between the application and the database management system. |
| CoocResults | Pops up when the "Corresponding A-terms" and "Corresponding B-terms" buttons are clicked to show the MeSH terms that have co-occurred with the selected terms. |
| DBFacade | This class implements the Facade Pattern as all the database queries are posed there. |
| Done | Notifies the user when the requested process is completed. |
| EnterTerm | Pops up when the "Search Medline" button is clicked without entering a C-term. |
| Main | Starts the application and causes the main frame to pop up. |
| MainFrame | This class is where the main frame and the algorithms are implemented. |
| NoResults | Pops up if the semantic relation filters result in an empty ST_B or ST_A list, or if there are B-terms or A-terms found. |

Table 5-1. Classes and Their Main Functionalities

## *5.2 Comparison of Various Versions of the Software*

To evaluate our framework and study the impact of each of its main features, we have also implemented three versions for comparison purposes. These developments also provide a basis to compare our work with existing systems. More specifically, our

intention is to evaluate and analyse how providing interactivity and using the pipes and filters architecture affect the performance of the system by comparing various aspects, such as speed, in the case of the three different versions.

1. The first version implements the Bio-SbKDS algorithm.
2. The second version uses pipes and filters to create and execute the different processes, but it is non-interactive.
3. The third version is ExaminMED, which uses pipes and filters and also supports extensive user interactions.

In the first version, which implements the Bio-SbKDS algorithm, all the B-terms that correspond to the input C-term are computed before any A-term is searched for. For the input *c*, the algorithm first generates the semantic types ST_B and ST_A, which are used to prune the results and are based on the semantic relations selected by the user, and then searches in MEDLINE for those B-terms that have co-occurred with *c* and have passed the semantic filters. Once all such B-terms are found, Bio-SbKDS uses each B-term *b* and performs a search in MEDLINE for A-terms that have co-occurred with *b* and whose semantic type is listed in ST_A, the corresponding semantic type filter. We remark that Bio-SbKDS has only one thread of control and the steps are executed sequentially. That is, there is no notion of concurrency, even though there are steps that are somewhat independent and could run concurrently. This is why all B-terms are computed before it begins its search for any A-term. Also, Bio-SbKDS is fully automated and hence does not allow any human intervention during processing other than the user input at the beginning. This in particular means that the user does not have the option or opportunity to add or remove B-terms. Another limitation of Bio-SbKDS is that it does not provide

filtering option to users to influence the processing of the system. Our first implementation, which simulates Bio-SbKDS, does not need to keep track of which B-terms co-occurred with which A-terms. It suffices to only store the number of B-terms that co-occurred with each A-term. For example, for the A-term *Cortisone*, we only store the value 3, as this A-term has co-occurred with 3 terms in the B-terms' list. Our first version is simpler also due to its sequential nature and limited functionalities. Accordingly, its interface is slightly different and simpler, in fact. This interface is shown in Figure 5-1. Unlike ExaminMED, Bio-SbKDS does not provide concurrency, interactivity, extensibility, and flexibility with regard to filtering techniques, and does not support combination of filters through combining search results. As can be seen in Figure 5-1, there is no need for the "Add" or "Remove" buttons since Bio-SbKDS does not support interactivity. Also, there is no option for the user to choose the desired filters. It does not store the results in a database. The option of combining search results and hence supporting more complex filters is not available in Bio-SbKDS, and hence not present in our first version.

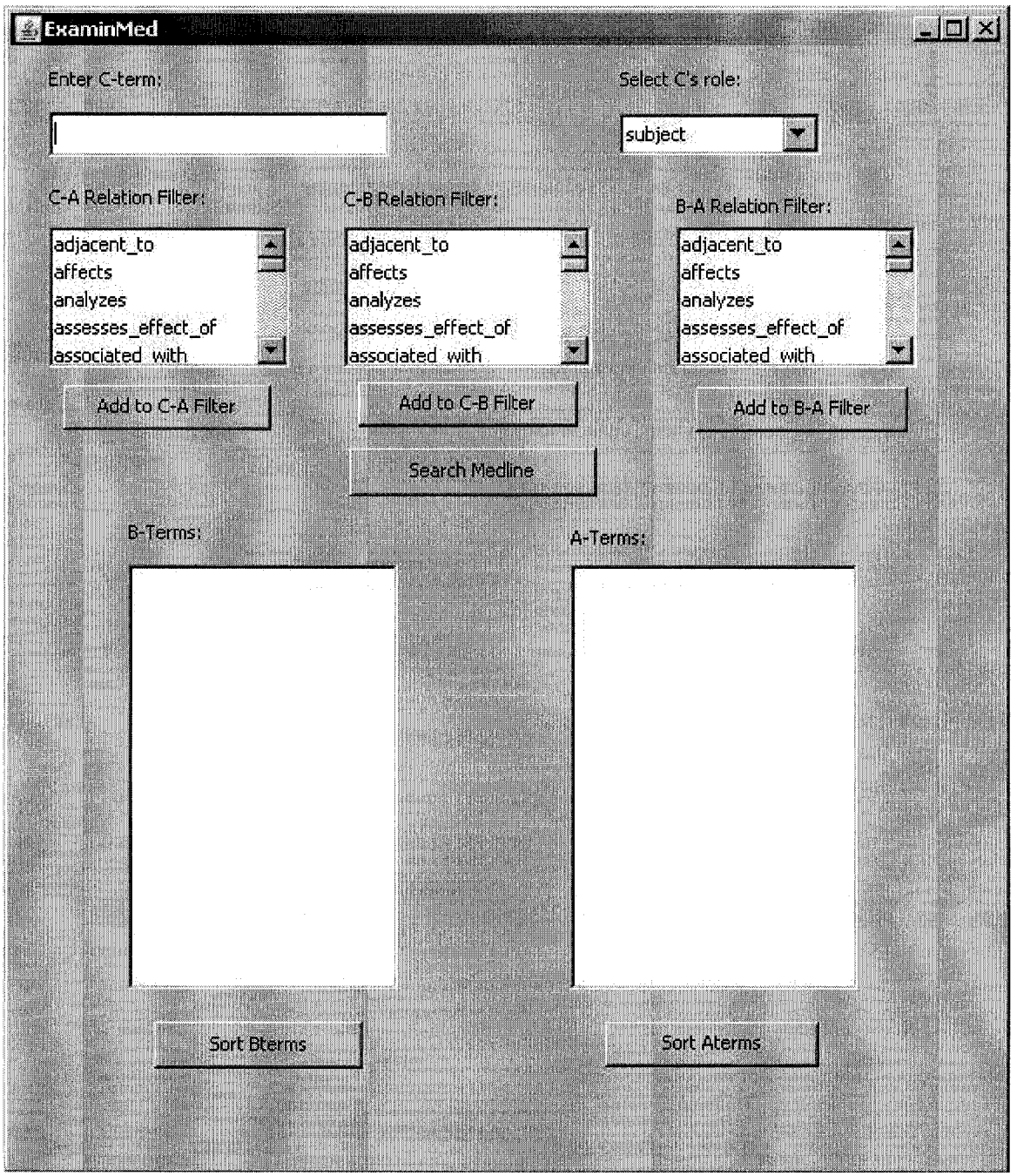


Figure 5-1. GUI of Versions 1 and 2 of the Prototype

The second version of the implementation makes use of the pipes and filters architecture, but is still non-interactive. Unlike our first implementation, this version allows concurrency by creating multiple threads. This implementation includes four filters,

which are connected through the pipes. The filters consist of a co-occurrence filter and a semantic filter for B-terms, and a co-occurrence filter and a semantic filter for A-terms. Once a B-term passes both filters, it is searched against MEDLINE to find its corresponding A-terms. Meanwhile, more B-terms are retrieved. This version is not interactive and hence the user may not add or remove terms, choose the filters, or combine search results. Its interface is similar to that of the first version, shown in Figure 5-1.

The third version of the implementation is that of ExaminMED, described in detail in the previous chapters. This implementation follows the pipes and filters architecture, supports interactivity, and incorporates multi-threading and concurrency. Since we found it useful to present the corresponding B-terms and A-terms to the user, we had to keep track of this information. To store this information, we used a two-dimensional array, which we called ABTerms. Each column in ABTerms is dedicated to a B-term; the top entry in this column contains the B-term, and each of the other entries in this column records the corresponding A-terms.

In terms of results, all three implementations produce identical results. However, they differ in execution times due to their different functionalities and features. Our goal in this section is to study, analyze, and compare the running time of each version to verify

(a) whether the concurrency provided by the pipes and filters architecture results in efficiency and reduces execution time, and

(b) how the overhead of providing interactivity affects execution time.

For this, we used two C-terms, *Raynaud Disease* and *Migraine Disorders,* as inputs to these three versions. Table 5-2 presents different execution times for these versions in various scenarios for the aforementioned C-terms. The different scenarios include (1) simple architecture or using pipe and filters, (2) interactive or non-interactive, and (3) using semantic filters or not using them. The first row in this table represents different scenarios and versions for which we measured the run times. The three left-most columns show the time measured when using semantic filtering. When semantic filtering was included, approximately 0.93 minutes of the reported runtimes was spent to calculate the ST_B and ST_A lists, the allowable semantic types of B-terms and A-terms.

When semantic filtering is not selected, the prototype runs significantly faster. This is also apparent in the runtimes demonstrated in Table 5-2. There are three main reasons for this noticeable difference in execution time. The first obvious reason is that the system is not required in this case to use any semantic type, as there was no semantic filtering selected. The second reason for increased efficiency in this case is that the system does not need to connect to UMLSKS to retrieve the semantic types of the terms stored in the database, since there is no need to prune terms according to their semantic types. Finally, the system avoids the necessity to check the semantic type of each term since any B-term or A-term is displayed to the user, no matter of what semantic type it is.

Table 5-2 indicates that the first version runs faster compared to the third version. It is important to note that this is not due to the overhead of incorporating pipes and filters in the third version. Rather, it is because unlike the third version, the first version is non-interactive and hence simpler, and provides less functionalities, all of which result in lower execution time. In order to verify our claim, we used the second version to perform

the same search operations. Recall that the second version is non-interactive and uses pipes and filters. Using the same C-terms as inputs to the second version, the executions time measured were less than corresponding search by the first version. These experiments confirm our idea that using pipes and filters can results in increased efficiency of our system prototype. The reduced execution time is mainly due to the concurrent execution of multiple threads, which is provided by pipes and filters. The system begins producing A-terms once a B-term is passed through both filters. Therefore, B-terms and A-terms are produced concurrently, which makes the system more efficient.

| | Version 1/ Semantic Filtering | Version 2/ Semantic Filtering | Version 3/ Semantic Filtering | Version 3/ no Semantic Filtering |
|---|---|---|---|---|
| Raynaud Disease | 2.38 | 1.84 | 2.60 | 0.23 |
| Migraine Disorders | 2.73 | 2.12 | 2.93 | 0.39 |

Table 5-2. Runtimes (in minutes) for Different Versions of our Implementation

Table 5-2 also shows that the search for *Migraine Disorders* in each case took longer to execute compared to that of *Raynaud Disease*. This is because *Migraine Disorders* co-occurred with more B-terms. Specifically, when semantic filtering was not applied, searching *Migraine Disorders* resulted in 20 B-terms and 656 A-terms, while *Raynaud Disease* occurred with only 13 B-terms and 563 A-terms.

Applying semantic filters resulted in 5 B-terms and 41 A-terms for *Migraine Disorders.* Searching for *Raynaud Disease* in the same case resulted in 5 B-terms and 51 A-terms. To explain why the search for *Migraine Disorders* still took longer to perform, it suffices to note that even though the number of B-terms is the same and the number of A-terms

for *Migraine Disorders* are fewer than that of *Raynaud Disease*, the system still has to process a larger number of B-terms for *Migraine Disorders*, i.e., 20 B-terms versus 13 B-terms. In general, in all three versions, a larger number of B-terms results in longer execution time, since more data needs to be processed.

We expect that as the database grows, the second and third versions of our implementation will outperform the first version. This is because the benefits of concurrency will be more visible with larger amounts of data. The first version has only one thread of control and processes the steps sequentially. For data of limited size, this is acceptable. However, as the amount of data in the database grows, the time needed to sequentially process the data increases. Taking advantage of having multiple threads of control enhances the system performance, especially when the database is large.

## *5.3 Correctness and Usefulness*

In this section we discuss bases for correctness of our framework and its usefulness.

### 5.3.1 Correctness

To show correctness of our implementation, we need to ensure (1) correctness of the semantic types generated based on the user input, and (2) relevancy of the A-terms returned. As explained in the previous chapter, selecting semantic types explicitly is not required in our prototype to perform its operation. Instead, the user chooses the semantic relations between C and B, B and A, and C and A. The semantic relations are then used

by the system to generate the semantic types based on the information in the UMLS Semantic Network. We follow the Bio-SbKDS algorithm proposed by Yoo [22] as a basis for correctness of our system, described as follows. We used the same semantic relations in Bio-SbKDS and compared the results with ours.

For the C-term *Raynaud Disease*, whose semantic type is "Disease or Syndrome," we entered the following semantic relations as did Yoo:

C-A relation filter: *treats, prevents*

C-B relation filter: *process_of, result_of, manifestation_of, causes, affects*

B-A relation filter: *interacts_with, produces, complicates, affects*

Our implementation produced almost identical semantic types to that of Bio-SbKDS. Our system found 26 semantic types for B-terms and 14 for A-terms. These semantic types are shown in Table 5-3. Bio-SbKDS produced the same semantic types for A-terms. As for the semantic types of B-terms, it produced 25 of the 26 semantic types found by our system. This is because Bio-SbKDS eliminates "Disease or Syndrome" from ST_B, since it is the semantic type of the C-term. We remark that the algorithm in Bio-SbKDS excludes the semantic type of the C-term from being in the ST_B list. We took a conservative approach here and decided not to eliminate such semantic types, since after all, the user has a final say on which term to keep or eliminate.

In general, our first version which implements Bio-SbKDS produces the same semantic types as does Bio-SbKDS. This establishes the first basis for correctness of the semantic types produced by our system. Similar to Bio-SbKDS, our system finds semantic types through semantic relations, which are provided as input by the user. In other systems, however, users provide semantic types directly as input, rather than semantic relations.

| Allowable Semantic Types of B-terms (ST_B List) | Allowable Semantic Types of A-terms (ST_A List) |
|---|---|
| Cell Function | Antibiotic |
| Cell or Molecular Dysfunction | Pharmacologic Substance |
| Disease or Syndrome | Therapeutic or Preventive Procedure |
| Experimental Model of Disease | Chemical Viewed Functionally |
| Genetic Function | Biologically Active Substance |
| Mental Process | Biomedical or Dental Material |
| Mental or Behavioral Dysfunction | Enzyme |
| Molecular Function | Hazardous or Poisonous Substance |
| Neoplastic Process | Hormone |
| Organ or Tissue Function | Immunologic Factor |
| Organism Function | Indicator, Reagent, or Diagnostic Aid |
| Pathologic Function | Neuroreactive Substance or Biogenic Amine |
| Physiologic Function | Receptor |
| Acquired Abnormality | Vitamin |
| Congenital Abnormality | |
| Amino Acid, Peptide, or Protein | |
| Carbohydrate | |
| Chemical Viewed Structurally | |
| Eicosanoid | |
| Element, Ion, or Isotope | |
| Inorganic Chemical | |
| Lipid | |
| Nucleic Acid, Nucleoside, or Nucleotide | |
| Organic Chemical | |
| Organophosphorus Compound | |
| Steroid | |

Table 5-3. Semantic Types for B-terms and A-terms Produced by our Prototype

According to Yoo, the semantic types generated by Bio-SbKDS, and thus our prototype, are similar to those selected by experts.

To test the correctness of their systems, authors of similar work search in MEDLINE articles published prior to the discovery of certain knowledge, to check whether their systems could correctly identify the known links. For example, the connection between *Dietary Fish Oil* and *Raynaud Disease* was discovered in 1986. To investigate whether

the systems can discover this link through the ABC model, Bio-SbKDS searches the C-term *Raynaud Disease* in MEDLINE articles publish before 1986. The results show that *Fish Oil* is indeed among the top-ranked A-terms retrieved by the system.

Since our database contains only a small portion of MEDLINE, we are not able to conduct the same kind of tests performed by Yoo and some other researchers. This is because many of the relevant terms which should ideally appear in our results do not reside in our local database, and thus will not be processed by our system. For example, a manual search which we conducted shows that our local database contains only one article whose indexed MeSH terms include *Fish Oils,* and that particular article has no other indexed terms. Therefore our system would have no way of connecting *Fish Oils* to the C-term *Raynaud Disease* through a B-term. Thus, in order to evaluate whether the A-terms retrieved by our system are indeed meaningful A-terms and are in fact related to the corresponding C-terms, we used PubMed and manually looked for the C-term accompanied with the top twenty A-terms retrieved by the system to verify whether the connection suspected by our system is meaningful. For example, in order to verify whether the connection between the C-term *Raynaud Disease* and the A-term *Cortisone* is a valid and known connection, we search *Raynaud Disease* and *Cortisone* together in PubMed. The result of this search indicated that MEDLINE contains five articles which have both terms as indexes and thus contain both terms together. Table 5-4 shows the result of manually entering, in PubMed, *Raynaud Disease* and the top twenty A-terms identified by our system. As the table indicates, 90% of the terms were relevant, as they had appeared at least once with *Raynaud Disease*. This analysis is yet another basis for concluding correctness of our system for identifying this valid link. Although this

connection was known, our system identified it as a new and unknown connection, because our local partial MEDLINE database did not contain any of the articles that contain the two terms together. Other A-terms retrieved by our system have also co-occurred with *Raynaud Disease* in numerous articles.

| A-term | Number of entries where A appeared with C in PubMed |
|---|---|
| Adrenal cortex hormones | 77 |
| Adrenocorticotropic hormone | 3 |
| Adrenalectomy | 9 |
| Hyaluronoglucosaminidase | 2 |
| Antacids | 4 |
| Arteries | 489 |
| Anti-bacterial agents | 50 |
| Blood proteins | 51 |
| Bone marrow | 31 |
| Castration | 27 |
| Cholinesterases | 2 |
| Cold | 847 |
| Dibenzylchlorethamine | 0 |
| Dicumarol | 2 |
| Electric stimulation therapy | 19 |
| Electrolytes | 4 |
| Gonadotropins | 10 |
| Heparin | 60 |
| Homeopathy | 0 |
| Hydantoins | 1 |

Table 5-4. Number of Co-occurrence Entries of Top-twenty A-terms with Raynaud Disease in PubMed

The B-terms and A-terms retrieved by ExaminMED conform to that of the first version, and thus to Bio-SBKDS. This is natural and expected. Although the queries posed to the database look different at first glance, they are essentially the same and thus, produce similar results, as expected. In the first version, we posed a query in which all the B-

terms are searched at the same time using the OR operator to merge the results. In ExaminMED, each B-term is searched in a separate query and the results are all displayed once they are retrieved. Therefore, the end results are similar because in SQL applying the OR operator to the B-terms in a single query is the same as applying it to the results of individual queries of each B-term.

### 5.3.2 Evaluation of Usefulness

The usefulness of the proposed software solution from the point of view of end users was not planned as a part this master's research. We note that a formal usability study in our context is an important future research. However, we informally discuss the usefulness of ExaminMED.

To informally evaluate usefulness of our prototype, we review the features presented in chapter 4 and see if they address the requirements defined in the outset, in section 1.2. It is important to evaluate whether or not ExaminMED addresses the needs of target users and considers their specific characteristics and limitations. Our target users for this system are doctors and clinical researcher with limited and valuable free time. Since there are a limited number of specialists available at hospitals and the number of patients they have to visit on a regular basis is high, the doctors are high on demand. Therefore they have a time constraint and it is important that the results be delivered to them as quickly as possible and the system can be used conveniently. In our design process, we took this fact into consideration and addressed this issue by allowing the results of each search to appear in the interface as soon as they are retrieved. This allows the user to reflect on the

retrieved B-terms and A-terms while the process continues to run in the background to produce the remaining related terms. Considering the large amount of data to be searched as a response of the user query, it is a very useful feature to provide partial results to the user as early as possible while the system is busy computing more results.

In addition, learning to use the system is not as time consuming and is fairly straightforward. In order for the users to be able to make use of ExaminMED, they need to comprehend the basics of the ABC model. Once the basics are understood, the fairly simple GUI is easy to use. This is especially important for our specific users who, as mentioned before, are busy and cannot afford to spend much time learning the software. This, however, may need a thorough usability test which is outside the scope of this research.

As for selection of semantic types, while we could allow users to directly choose the semantic types for B-terms and A-terms, due to the doctors' time constraint, we believe it would be more desirable for them to select among the 54 semantic relations rather than the 134 semantic types. Selecting items from smaller lists would be more affordable to the users. Also, we believe it is cognitively easier to think of semantic relations as opposed to exact semantic types. According to Yoo, the semantic types automatically produced by Bio-SbKDS, and thus by our prototype, are very similar to those chosen directly by experts in similar existing systems.

Also, the users are probably not computer experts and may have little knowledge about databases. It is thus important to provide a user friendly interface to query the database and display the results in a presentable way.

Furthermore, the flexibility provided by our system allows the user to experiment with different settings and supports users' interaction. For example, by allowing the user to select the desired semantic types, we eliminate restrictions and allow the user to get involved easily and freely.

Also, the concurrency which ExaminMED provides is useful upon noting the vast amount of information that needs to be processed. As the number of terms grows, the need for concurrency becomes more apparent since it significantly improves performance in terms of runtime.

Finally, after conducting research on the ABC model and its various extensions, we found it useful to build a framework which is not hard-coded and can be extended easily so that new filters, features, and functionalities can be added in a straight-forward manner and without much modification to the rest of the implementation.

# CHAPTER 6

# CONCLUSION AND FUTURE WORK

## *6.1 Summary and Conclusion*

In this work, we studied the issue of discovering Undiscovered Public Knowledge (UPK) from medical literature by suggesting an interactive, flexible, extensible, and concurrent framework of Swanson's ABC model based on a semantic notion. We designed and developed an interactive prototype, ExaminMED, which makes use of the pipes and filters software architecture to search the MEDLINE database to discover previously unknown knowledge.

### 6.1.1 Interactive, Flexible, Extensible, and Concurrent UPK Discovery

The main goal in this research was to design a flexible, extensible, and interactive framework for UPK discovery using the ABC model. We identified the need for a software framework since the ABC model was extended by various researchers, all of whom developed software from scratch. An extensible software framework allows software developers to reuse existing components of the software and simply incorporate the new idea without needing to make much modification to the rest of the software.

To demonstrate the abilities of the framework, we followed the Bio-SbKDS algorithm to create a semantic-based version of Swanson's ABC model. The semantic knowledge incorporated in our prototype is extracted from two well-known ontologies, MeSH and

UMLS. Unlike Bio-SbKDS, which is completely automated, ExaminMED is semi-automated and interactive. We provide various features and functionalities to our users to facilitate the discovery process. Since the task of discovering UPK is exploratory, it is important to allow the users to interact with the prototype. ExaminMED is flexible enough to allow the user the freedom to perform exploratory tasks such as choosing desired filters, combining the results of various searches, and experimenting with different combinations of terms.

We used the pipes and filters architecture in the design of our prototype system to benefit from the advantages that come naturally with this architecture, including flexibility, extensibility, concurrency, and possibly parallel processing in future implementation. Currently, we have implemented two co-occurrence filters and two semantic filters for pruning the B-terms and A-terms. The use of the pipes and filters architecture allows easy addition of new filters and/or modification of existing ones. Moreover, it allows users to combine or remove filters easily. This flexibility is very useful, considering the exploratory nature of UPK in the vast amount of medical literature. We verified correctness and usefulness of our proposed framework by comparing it to existing systems. Our experiments and analysis of the results indicate that our framework produces answers which conform to PubMed data. As further explained below, this is important upon noting that the database used in our implementation and experiments is just a small portion of the entire MEDLINE database.

As input to the prototype system, the user enters a C-term and three semantic relation filters. Our system will then perform a search in the MEDLINE database, looking for concepts that have not co-occurred with the given C-term but are potentially related to it

through a third concept, a B-term. Analysis of the newly discovered concept relations may result in generation of new hypotheses by the user. If the hypothesis proves to be valid after clinical investigation or verification by an expert, then we say new knowledge is discovered. This capability to hypothesize is important noting the huge amount of information in the medicine and health sciences domain. Ontologies and semantic knowledge can also improve the search quality by retrieving more meaningful concepts and pruning a large number of terms present in the database. Our prototype allows the user to select and use suitable filters for the UPK discovery process. We also store, in the database, the results of the queries and searches. This information is used by our system to allow users to explore the results using the database technology and also allows formulation of complex queries using SQL by combining the results of various C-terms, if desired.

One of our goals in this research was to build a working prototype to illustrate the proposed ideas, as opposed to developing a complete software product. We only used a small portion of MEDLINE and illustrated sample queries to explore the search results. Since the database is not complete, our prototype may retrieve A-terms that are already known in some MEDLINE articles to be correlated to the input C-term, however, because those articles are not in our small sample database, our framework identified the A-C connection as a new one, which in itself was interesting for us to re-discover. In order to test correctness of the results, we manually search the connection of the C-term with top-ranked A-terms in PubMed, which is the web-based interface to the entire MEDLINE. Our running prototype was successful in discovering some of these "known" relations, indicated by co-occurrence of the C-term and A-terms in MEDLINE. In other words, the

A-terms retrieved by our prototype are meaningful in that they were known to be related to the C-term in the literature.

We have also compared our results with those produced by similar systems. We found that for every relevant term discovered by other existing software, our prototype was also able to identify and retrieve the term if it was present in our local, incomplete database. As per semantic filtering, we compared our semantic type filters with those implemented in Bio-SbKDS. Our experiments and results indicated that ExaminMED generated all the semantic types produced by Bio-SbKDS. This is also another indication of correctness of our implementation since it implements the same algorithm for semantic filter generation used in Bio-SbKDS.

In our work, we used the BioText tool for parsing the MEDLINE database, available to licensees in XML format. We also used the tool to load the parsed entries into our local database, for which we used the IBM DB2, a relational database management system, to store and retrieve the transformed MEDLINE data. Each medical article in MEDLINE is indexed by domain experts to reflect the main concepts addressed by the article. As part of query processing and search, ExaminMED uses these indexes to find co-occurring terms more efficiently. After the semantic types are verified to match the semantic filters, the concepts which have not directly co-occurred with the input C-term but indirectly through a third item are found and presented to the user through a simple user interface we developed. The user may then explore the results further and suggest new hypotheses. Our experiments and interactions with the developed system prototype indicate that the proposed ideas are promising and useful in practice. Also, due to its use of pipes and filters architecture, the proposed framework is flexible and extensible, important in the

concerned application for being exploratory in nature. The particular choice of pipes and filters architectural design allows efficient implementation through concurrency and possibly parallel processing, if desired. The advantages of using the pipes and filters architecture in our design conformed to the requirements which we identified for such systems. Therefore, by using this architecture, we have met the requirements of our prototype.

## *6.2 Future Work*

There is much more that can be done in this endeavour to improve and enhance our proposed framework for discovery of UPK. In this section we highlight some potential future work related to this research, including result-sharing among experts, building and employing more specialized ontologies, and providing parallelism.

### 6.2.1 Sharing Results among Experts

Currently, our prototype assumes only one user. This can be extended to a network in which multiple users and experts can access the software, interact with it, and share their results and experiences. By allowing this type of information sharing, the hypothesis generation process may improve since an expert may think differently than another, and sharing their expertise and ideas may be of benefit to them. Each user may use different filters, add or remove different terms, and combine the results of different searches in

his/her own unique way. Sharing this information and allowing users to discuss different decisions made when interacting with the system may prove to be useful for knowledge discovery process. Experts may benefit from other experts' ideas and experiences.

### 6.2.2 Using Specialized Ontologies instead of MeSH and UMLS

Currently, our system benefits from two existing and well-known ontologies, namely MeSH and UMLS. While these are fairly large and relatively complete ontologies, they do not contain highly domain-specific terms for various medical specialties. Making use of a more specialized ontology may be useful when experts are interested in discovering unknown information in that particular expert domain. Building domain-specific ontologies, which can be queried and used instead of MeSH and UMLS, can boost discovery opportunities in medical domain, resulting in far better and more economical health care systems and human well-being.

### 6.2.3 Parallel Processing

As mentioned previously, in our current implementation different steps of the algorithm are processes created as independent threads, when possible, and executed. When multiple processors are available, multi-threading can turn into multi-processing, when there are multiple resources. Parallel processing would make the discovery process in our context faster and more efficient. Each processor can execute the algorithm on specific chunks of the MEDLINE database and the results can be merged when retrieved.

Considering the size of MEDLINE, dividing this data among various processes can improve performance significantly. More work is required before this can be realized, however, the point to emphasis here is that the ideas proposed and the design decisions made in our development in this research make parallel processing a possible natural choice for further development of this work in particular when extending it to a large network of expert users and labs.

# LIST OF REFERENCES


1. Aronson, A.R., "MetaMap: Mapping Text to the UMLS Metathesaurus," 2006.
2. Bashiiui, "Pipes-and-Filters Pattern - An architectural design pattern," http://www.codeproject.com/KB/architecture/PipesAndFilters.aspx, February 2006.
3. "BioText Software for Download," University of California, Berkeley, http://biotext.berkeley.edu/software.html, 2007.
4. Cunningham, H. C., "Pipes and Filters Architectural Pattern," http://ftp.cs.olemiss.edu/~hcc/softArch/notes/pipes.html, March 2004.
5. Han, J., Kamber, M., "Data Mining Concepts and Techniques," 2000, pp.1.
6. Life Sciences Library, McGill University, "MeSH," http://www.health.library.mcgill.ca/help/guides/biomed/mesh/, February 2004.
7. National Library of Medicine, "UMLS 2004AB Documentation," http://www.nlm.nih.gov/research/umls/META3_current_semantic_types.html, November 2003.
8. National Library of Medicine, "About the UMLS Resources," http://www.nlm.nih.gov/research/umls/about_umls.html, May 2006.
9. National Library of Medicine, "Unified Medical Language System," http://www.nlm.nih.gov/research/umls/META3_Figure_1.html, May 2006.
10. National Library of Medicine, "Medical Subject Headings," http://www.nlm.nih.gov/mesh/ , October 2007.
11. National Library of Medicine, "2007 MEDLINE/PubMed Baseline Distribution," http://www.nlm.nih.gov/bsd/licensee/2007_stats/baseline_doc.html, December 2006.
12. National Library of Medicine, "Unified Medical Language System Knowledge Source Server (UMLSKS) Fact Sheet," http://umlsks.nlm.nih.gov/kss/servlet/Turbine/template/admin%2Cuser%2CFactSheet.vm, May 2007.

13. National Library of Medicine, "Mesh Descriptor Data," http://www.nlm.nih.gov/cgi/mesh/2007/MB_cgi, 2007.
14. Open Biomedical Ontologies, http://obo.sourceforge.net/main.html.
15. Oliver D.E., Bhalotia G., Schwartz A.S., Altman R.B., Hearst M.A., "Tools for Loading MEDLINE into a Local Relational Database," BMC Bioinformatics, 5, pp. 146–157, October 2004.
16. Srinivasan, P., "Text Mining: Generating hypotheses from MEDLINE," Journal of the American Society for Information Science, Vol. 55, No. 5, pp. 396-413, March 2004.
17. Swanson, DR., "Undiscovered Public Knowledge," Library Quarterly, Vol 56, No 2, pp. 103-118, 1986.
18. Swanson, DR., "Medical Literature as a Potential Source of New Knowledge," Bull Med Libr Assoc, Vol 78, No 1, January 1990.
19. Swanson, DR., Smalheiser, NR., "An Interactive System for Finding Complementary Literatures: A Stimulus to Scientific Discovery," Artificial Intelligence, Vol 91, No 2, April 1997, pp. 183-203.
20. Weeber, et al, "Using concepts in literature-based discovery: Simulating Swanson's Raynaud-Fish Oil and Migraine-Magnesium Discoveries," Journal of the American Society for Information Science and Technology, Vol. 52, No. 7, pp. 548-557, 2001.
21. Wikipedia, "Ontology (Computer Science)," http://en.wikipedia.org/wiki/Ontology_%28computer_science%29, 2007.
22. Yoo, I., "Semantic Text Mining and its Application in Biomedical Domain," PhD Thesis, Drexel University, July 2006.