

AMERICAN UNIVERSITY OF BEIRUT

Semi-Automatic Annotator for Medical NLP  
Applications

by  
Mohamed Naji Sabra

A thesis  
submitted in partial fulfillment of the requirements  
for the degree of Masters in Engineering  
to the Department of Electrical and Computer Engineering  
of the Faculty of Engineering and Architecture  
at the American University of Beirut

Beirut, Lebanon  
August 2015

AMERICAN UNIVERSITY OF BEIRUT

Semi-Automatic Annotator for Medical NLP  
Applications

by  
Mohamed Naji Sabra

Approved by:

---

Dr. Fadi Zaraket, Assistant Professor  
Electrical and Computer Engineering

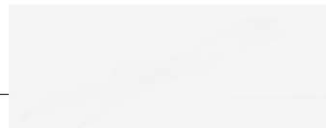
Advisor



---

Dr. Mariette Awad, Associate Professor  
Electrical and Computer Engineering

Member of Committee



---

Dr. Rouwaida Kanj, Assistant Professor  
Electrical and Computer Engineering

Member of Committee



Date of thesis defense: August 3, 2015



# An Abstract of the Thesis of

Mohamed Sabra for Master of Engineering  
Major: Electrical and Computer Engineering

Title: Semi-Automatic Annotator for Medical NLP Applications

With the expansion of scientific and social media, a wealth of online information resources has accumulated as free text including articles, studies, and social blogs. Mining, standardization, and extraction of information from these resources brings upon novel approaches for data analysis and knowledge discovery; particularly from domain specific large text corpora.

Key to this is annotated corpora. Supervised algorithms for machine learning need them for training. Unsupervised algorithms need them for testing and evaluation. Manual annotation is expensive specially in expert domains such as medicine.

This thesis presents a Semi-Automatic Annotator for Medical NLP Applications (SAMNA) . SAMNA takes a large corpus, a list of labels, a list of terms associated with each label, and lists of rules associated with labels and terms. SAMNA annotates the corpora words that match the corresponding terms and rules. It also uses distributional similarity to discover novel annotations. In addition, it provides the annotating scholar with an intuitive, friendly and efficient interface to navigate and edit the annotations.

We used SAMNA in several medical NLP applications to annotate protein sets in medical articles related to specific diseases such as stroke, spinal chord injuries, and Alzheimer. The graph theory based analysis of the corpora annotated with SAMNA led to discoveries on interest to medical experts.

SAMNA can also be applied in systems review, as well as other annotation domains.

# Contents

<b>Abstract</b>	<b>4</b>
<b>Contents</b>	<b>i</b>
<b>List of Figures</b>	<b>iii</b>
<b>List of Tables</b>	<b>v</b>
<b>1 Abstract</b>	<b>1</b>
<b>2 Introduction</b>	<b>2</b>
<b>3 Preliminaries</b>	<b>7</b>
3.1 Annotations . . . . .	7
3.2 Distributional Similarity . . . . .	7
3.3 Medical NLP . . . . .	8
<b>4 Motivation</b>	<b>10</b>
<b>5 Semi Automatic Annotator</b>	<b>13</b>
5.1 SAMNA File Extraction . . . . .	13
5.2 SAMNA Navigation . . . . .	14
5.3 SAMNA Data Model . . . . .	15
5.4 SAMNA Annotation . . . . .	15
5.5 SAMNA User Editing . . . . .	16
5.6 SAMNA Analysis . . . . .	16
<b>6 Method</b>	<b>18</b>
6.1 Direct String Matching . . . . .	18
6.2 knowledge Base Rules . . . . .	18
6.3 Distributional Similarity Algorithm . . . . .	20
6.4 Choosing $\delta_1$ and $\delta_2$ . . . . .	23

<b>7</b>	<b>Implementation</b>	<b>28</b>
7.1	File Reader . . . . .	28
7.2	Highlighter . . . . .	32
7.3	Annotation . . . . .	33
<b>8</b>	<b>Experimental results</b>	<b>35</b>
8.1	Stroke Case Study . . . . .	35
8.1.1	Method . . . . .	36
8.1.2	Results . . . . .	39
8.1.3	Conclusion . . . . .	43
8.2	Case study 2 spinal cord injury . . . . .	44
8.2.1	Materials and Methods . . . . .	45
8.2.2	Results . . . . .	46
8.2.3	Conclusion . . . . .	47
<b>9</b>	<b>Related Work</b>	<b>48</b>
<b>10</b>	<b>Conclusion and Future Work</b>	<b>50</b>
	<b>Bibliography</b>	<b>51</b>
	<b>Appendix A Tutorial</b>	<b>55</b>
A.1	How to run the tool . . . . .	55
A.2	Tool Prerequisite . . . . .	55
A.3	Tool Features . . . . .	56
A.3.1	Loading frame . . . . .	56
A.3.2	Main frame . . . . .	57
A.4	Loading Abstracts . . . . .	58
A.4.1	Loading Abstracts from an XML file . . . . .	59
A.4.2	Loading Serialized Abstracts . . . . .	60
A.4.3	Loading Abstracts from TXT and CSV file . . . . .	60
A.5	Moving between Abstracts . . . . .	60
A.6	Adding Annotation . . . . .	61
A.7	Removing Annotation . . . . .	61
A.8	Apply expert Rules . . . . .	62
A.9	Distributional Similarity suggestion . . . . .	62
A.10	Adding New Labels . . . . .	62
A.11	Annotation distribution . . . . .	63
A.12	Exclude Abstract . . . . .	63
A.13	Undo Actions . . . . .	63

# List of Figures

6.1	String matching algorithm . . . . .	19
6.2	Create hash table from label spread sheet . . . . .	19
6.3	Get similarity for 1 word protein . . . . .	21
6.4	Get collocation for 1 word protein . . . . .	21
6.5	similarity and collocation computation . . . . .	22
6.6	Get the 1 word suggested similarity List . . . . .	24
6.7	distributional similarity algorithm . . . . .	25
6.8	Get the multi word similarity List . . . . .	26
6.9	Precision alternating $\delta_1$ and $\delta_2$ . . . . .	27
8.1	overview of data curation and processing . . . . .	36
8.2	captured set of accessions . . . . .	37
8.3	articles selection process . . . . .	38
8.4	clustering of proteins in the BIMP expressed by brain tissue . . . . .	39
8.5	Venn Diagram of the distribution of BIMP proteins on different significantly enriched KEGG pathways . . . . .	40
8.6	Markov clustering of the network of BII . . . . .	41
8.7	Markov clustering of the network of BII . . . . .	42
8.8	Graph of protein-protein interactions among components of major KEGG enriched pathways . . . . .	43
8.9	Rich-club organization in the BII . . . . .	43
8.10	Estrogen targets within the BIMP . . . . .	44
8.11	SPINAL CORD . . . . .	46
A.1	Choosing work space . . . . .	56
A.2	SAMNA Loading frame . . . . .	57
A.3	SAMNA main frame . . . . .	57
A.4	loading Abstract Picture . . . . .	59
A.5	adding protein . . . . .	61
A.6	suggestion base on the distributional similarity algorithm . . . . .	62
A.7	Adding new label . . . . .	63
A.8	annotation distribution . . . . .	63
A.9	removing abstract . . . . .	64

A.10 undo an action . . . . .	64
-------------------------------	----



# List of Tables

6.1	precession while alternating $\delta_1$ and $\delta_2$ . . . . .	27
-----	--	----



# Chapter 1

## Abstract

With the expansion of scientific and social media, a wealth of online information resources has accumulated as free text including articles, studies, and social blogs. Mining, standardization, and extraction of information from these resources brings upon novel approaches for data analysis and knowledge discovery; particularly from domain specific large text corpora.

Key to this is annotated corpora. Supervised algorithms for machine learning need them for training. Unsupervised algorithms need them for testing and evaluation. Manual annotation is expensive specially in expert domains such as medicine.

This thesis presents a Semi-Automatic Annotator for Medical NLP Applications (SAMNA) . SAMNA takes a large corpus, a list of labels, a list of terms associated with each label, and lists of rules associated with labels and terms. SAMNA annotates the corpora words that match the corresponding terms and rules. It also uses distributional similarity to discover novel annotations. In addition, it provides the annotating scholar with an intuitive, friendly and efficient interface to navigate and edit the annotations.

We used SAMNA in several medical NLP applications to annotate protein sets in medical articles related to specific diseases such as stroke, spinal chord injuries, and Alzheimer. The graph theory based analysis of the corpora annotated with SAMNA led to discoveries on interest to medical experts.

SAMNA can also be applied in systems review, as well as other annotation domains.

# Chapter 2

## Introduction

With the expansion of scientific and social media, a wealth of information has accumulated as free text within online resources including articles, studies, and social blogs. Mining, standardization, and extraction of this information would bring upon new approaches for analysis and discovery of knowledge within large text corpora specific to certain disciplines. Natural language processing (NLP) tasks deal with text corpora, and map these corpora to output domains. Those output domains differ according to the NLP task. Example output domains are (1) another language in case of a machine translation (MT) task, (2) word classification in case of part of speech (POS) annotation task, and (3) categories in an information extraction (IE) task.

There are two NLP approaches: a supervised, and an unsupervised approach. Both NLP approaches need a reference text corpus annotated with correct data alongside with a testing text corpus to evaluate the accuracy of the NLP technique. A supervised approach will also require a training text corpus that is annotated in a manual or standard manner to learn a computational model [1].

There is an ongoing interest in applying NLP information and relational extraction tasks to biomedical literature and texts. Several tools have been developed in the field of medical and biological NLP to handle different types of corpora [2] including:

- Electronic medical records (EMR),
- Medical literature databases such as PubMed and Scopus,
- Genia Event Extraction (GE),
- Cancer Genetics (CG),
- Pathway Curation (PC),
- Gene Regulation Ontology (GRO),

- Gene Regulation Network in Bacteria (GRN), and
- Bacteria Biotopes (BB).

Several annotators such as Knowtator [3], Brat [4], WordFreak [5], MMAX2 [6], Callisto [7], and Turku Event Extraction System TEES [8] are already present and were utilized in some biomedical applications such as the use of TEES in curation of the Gene Regulatory Network in Bacteria [2]. However, these annotators may not be optimal to all tasks of biomedical NLP, and optimized models of these annotators would better fit specific biomedical NLP tasks.

We are particularly interested in devising new biomedical NLP tools to convert the massive literature on complex diseases and conditions into a resource that fosters a better understanding of the disease processes and therapy. The biochemical basis of disease can be reduced into disturbances in gene products (such as proteins) that would lead to abnormal cellular biology leading to pathology. Proteins and their relation to diseases have become important in recent medical research especially that proteins are the final biochemical effectors and the most interesting targets for therapeutic intervention.

In the field of protein to disease relations, there is a huge number of articles that report protein changes during or after a certain disease. A wide range of proteins is reported by these studies depending on the author's interest, the particular aspect of disease studied or the animal model. However, the collective contribution of the entire spectrum of protein changes is the determinant of overall disease pathology. This spectrum of protein changes cannot be characterized by one or few studies and requires cumulative evidence over decades of scientific research. Therefore, efforts to extract and curate the spectrum of protein changes in disease from the large number of scientific reports would allow for both a global and detailed analysis of molecular architecture of diseases that includes the full network of proteins interactions responsible for disease phenotype and symptoms.

**the aim of this work is to use semi-automatic annotation and biomedical NLP algorithms to provide a model for curation and construction of a molecular architecture of pathological and disease conditions.** The resulting model is particularly significant in the context of complex diseases where a large number of proteins contribute to the disease processes such as Alzheimer's disease, stroke, myocardial infarct, and cancer metastasis.

Articles reporting protein changes after disease are stored in online biomedical databases like PubMed and SCOPUS. The databases allow using a special API for the retrieval of the data in a predefined format such as EXtensible Markup

Language (XML) and comma-separated values (CSV) depending on a search key specific to the disease or disease process of interest.

Each article presents evidence of the role of one or more proteins in activating a pathway of one or more diseases. To be able to treat a disease, medical researchers are interested in looking at the comprehensive picture.

A search in PubMed and Scopus returns 83,101 articles related to the stroke disease. This magnitude of literature is hard to analyze manually and automated NLP tools that help in the selection of the relevant articles are needed.

Existing annotators fall short of performing this task because:

1. Most of the existing annotators need a professional software developer to deploy and run them, which is hard for a protein or disease specialist.
2. Some of those annotators could not handle large file sizes such as 100 MB, while the data about Brain Ischemia is a 688 MB.
3. They cannot parse XML and CSV files specific to biomedical indexing databases.
4. They do not generate needed statistics from the annotated data (such as frequency and co-occurrence of the annotated terms).
5. They are single document based, and cross-document analysis is the target of this work.

Therefore, we need an annotator that allow for cross-document annotation surpasses all the above shortcoming and has the following abilities

- It should be user friendly and easy to install and use.
- It should parse large files and extract important data from files such as titles, abstracts, ids and dates.
- It should be able to display documentation aggregated across all articles and files, and allows for inspecting each one alone with its important components.
- It should allow the intervention of the specialist to edit and redo the automatic annotations including adding or deleting an annotation.
- It should be able to create statistical metrics that describe the annotations and their occurrence while performing the NLP task.
- It should be able to automatically suggest potential hits within biomedical texts with high accuracy.

We propose to build a Semi-Automatic Annotator for Medical NLP Applications (SAMNA), which allows the user to load a large amount of data including abstracts and titles of published articles that discuss a specific disease of interest. In addition to the required functionalities, SAMNA provides the following:

- SAMNA visualizes the components of articles such as title, abstract, id and date with color sensitive annotations, that highlight occurrences of terms of interest to the NLP task in the text.
- SAMNA takes expert rules from the specialist that specify features of the target annotations. It uses these rules to annotate additional terms not defined previously in the database. For example  $\forall P_i \in \text{list of protein } P;$   $P_i[1-9][1-9]^*$  is a protein. If XYZ is a protein in the database, XYZ3 will be annotated as protein.
- SAMNA implements a distributional similarity algorithm using DISCO [9] to find new annotations.
- SAMNA saves a result file holding the important statistics of annotation occurrences and frequencies for each annotation. In this way SAMNA provides a full protein spectrum of diseases for further analysis by biomedical investigators.

We evaluated SAMNA and used it with the following applications:

### **Stroke (or brain ischemia)**

We used SAMNA to construct, annotate and curate a brain ischemia meta-proteome that reflects the spectrum of protein changes after brain infarct. We used systems biology databases including UniProt [10] for protein accession retrieval and STRING [11] for protein interaction data, to construct the interaction network among the extracted proteins. Curation of the proteome allowed for dissecting the role of different biochemical pathways in the amplification of injury after stroke and revealed, with the help of graph theory algorithms, the presence of a rich-club organization in the brain ischemia interactome. The detected pathways and the rich-club provide information on the optimal approaches to target injury mechanisms after stroke [12].

### **Spinal Cord Injury**

We also used SAMNA to curate the proteome of spinal cord injury (SCI) and we were able to characterize the full molecular architecture of SCI. Analysis of the curated proteome using graph theory revealed a modular organization of disease process that are centered around a core of cell survival decision proteins. Those core proteins provide the best targets for therapy. [13]

## Cancer metastasis

Using the multi-label capability of SAMNA, we applied the same approach to study epithelial-to-mesenchymal (EMT) transition, the process by which cancer cells develop migrating and metastatic potential. SAMNA allowed to annotate for proteins, microRNAs, cancer subtypes, and drugs in order to detect the protein changes behind EMT and their relation to the cancer subtype, microRNA expression and to anti-cancer drugs.

We have also applied the same approach using SAMNA to other diseases such as Alzheimer's disease and Schizophrenia. And the analysis of the resulting graphs is in progress.

The rest of this thesis is organized as follows, chapter 2 introduces key medical NLP concepts, basic annotation terminology, and distributional similarity concepts that will be used in the thesis, chapter 3 discusses motivation behind the creation of SAMNA, chapter 4 introduces the functionality of the Semi Automatic Annotator, chapter 5 introduces the methods implemented in SAMNA, chapter 6 describes SAMNA technical implementation, chapter 7 introduces results and case studies, chapter 8 discusses related work, chapter 9 concludes with future suggestions, appendix A is a tutorial that passes through all functionalities of SAMNA.



# Chapter 3

## Preliminaries

This chapter introduces key medical NLP concepts, basic annotation terminology, and distributional similarity concepts that will be used in the thesis.

### 3.1 Annotations

We provide the following definitions and use them across the document. Given a set of documents and a set of labels where a document is a sequence of words, and a label is a designated word, we define the following.

**Definition 3.1.1 (Annotation).** An annotation is a tuple  $\langle w, pos, l \rangle$  where  $w$  is a word,  $pos$  identifies the document  $d$  and position of  $w$  in  $d$ , and  $l$  is a label.

**Definition 3.1.2 (Term).** A term is a word that has a label.

**Definition 3.1.3 (Legend).** A Legend is a set of display properties that allow visual identification of similarly labeled words.

**Definition 3.1.4 (Class).** A class, also known as annotation type is a couple  $\langle la, lg \rangle$  where  $la$  is a label and  $lg$  is a legend. We denote by  $class(la)$  the set of terms with label  $la$ .

**Definition 3.1.5 (Rule).** A rule is a regular expression that extends labels with wild cards of regular expression to match words in document (such as the asterisk  $*$ ). Example:  $\forall Pi \in \text{list of protein } P; Pi+''-''+[A-Z][A-Z]^*$  is a protein.

### 3.2 Distributional Similarity

The distributional similarity of two words  $w_1$  and  $w_2$  is a measure of how much those two words tend to occur in similar context.[\[14\]](#). Given a set of words  $w_1, w_2, \dots, w_n$ . The frequency vector of  $w_i$ ,  $i \in [1, \dots, n]$  is given by  $v = \langle C_1, C_2, \dots, C_n \rangle$  where  $C_i$  is the number of times  $w_i$  occurs around  $w$  within a window of words of size  $K$ .

**Definition 3.2.1 (Collocation).** A collocation of a word  $w$  is a set  $(C_1, R_1), (C_2, R_2), \dots, (C_n, R_n)$  where  $C_i$  is the word collocated to  $w$  within a window of words and  $R_i$  is the score of collocation. To compute the collocation score we used the following equation

$$g(w, w', r) = \log \frac{(f(w, r, w') - 0.95)f(*, r, *)}{f(w, r, *)f(*, r, w')}$$

Where  $w$  and  $w'$  stand for words and  $r$  for a window position (or a dependency relation, respectively), and  $f$  is the frequency of occurrence. To compute the collocation score we used DISCO[9] function which implement the above equation.

**Definition 3.2.2 (Similarity).** The similarity of a word  $w$  is a set  $(S_1, R_1), (S_2, R_2), \dots, (S_n, R_n)$  where  $S_i$  is the word similar to  $w$  and  $R_i$  is the score of similarity. To compute the similarity score we used the following equation

$$lin(w, w') = \frac{\sum_{p=1}^r \sum_{wi=1}^v \begin{cases} g(w, wi, p) + g(w', wi, p) & : g(w, wi, p) > 0 \text{ and } g(w', wi, p) > 0 \\ 0 & : \text{else} \end{cases}}{\sum_{p=1}^r \sum_{wi=1}^v (g(w, wi, p) + g(w', wi, p))}$$

To compute the similarity score we used DISCO[9] function which implement the above equation.

### 3.3 Medical NLP

Natural Language Processing (NLP) automates natural text understanding including entity and relational extraction. Medical NLP is interested in extracting entities such as protein names, symptoms, and treatments from medical texts such as publications and electronic medical records(EMR).

NLP uses three main approaches.

**Knowledge-based approach** The key elements of a knowledge-based system (KBS) are: [15].

1. Knowledge modules that contain expert information,
2. A knowledge base where the knowledge is stored,
3. A deduction engine for detecting solutions to problems from stored knowledge, and
4. A user interface that allows the model to accept user input and explain its processing and reasoning steps.

In our case, the publications are the knowledge modules. The predefined annotation and the expert rules are the knowledge base. The application of the rules, the distributional similarity algorithm, and the user feedback in the semiautomatic annotator process form the engine. Finally, SAMNA embedded all that in a user friendly interface.

**Empirical approach** Empirical approaches implement machine learning techniques to automatically extract linguistic knowledge from natural language data such as root detection, tokenization, part of speech tagging. The general trend for these approaches is to apply pattern recognition techniques such as K-nearest neighbour and support vector machines (SVM) to extract information from text [16].

These approaches originally don't require necessarily knowledge of the language rules.

**Hybrid approach** Recent techniques propose merging both approaches in which knowledge of the linguistic rules is used along with the empirical methods in order to boost the performance, and direct the learning process.

SAMNA uses both expert rules and statistical algorithms in addition to user feedback to extract information.

**Manual annotations** Given a text where we want to annotate terms that appertain to a label of interest, manual annotation is a techniques that allow the specialist to modify existing annotation by

1. adding / removing annotation.
2. adding / removing a label.

**Automatic annotation** Given a document where we want to annotate terms that appertain to a label of interest, automatic annotation is set of techniques that decide whether a word in that document is associated to a specific label, such as

- knowledge base rules which are some hard coded rules suggested by the specialist base on the knowledge of how the labels usually varies between each others, and
- distributional similarity algorithm that find annotation base on the similarity of the already founded annotation.

# Chapter 4

## Motivation

Medical studies typically review selected articles from a specific domain literature. The reviewer is limited to a humanly manageable number of articles, error prone, and hence not comprehensible. Collective knowledge embodied in all articles addressing a specific disease is hard to capture by sole human manual work. In particular, relations between proteins and specific diseases across all pubmed literature is of interest to medical scholars to shed some light on emerging and interacting disease pathway. Articles reporting on protein and disease co-occurrence exist in online biomedical databases like PubMed and SCOPUS. The databases allow using special API to retrieve the data in a predefined format, with a text base search mechanism.

The search mechanism works with exact string matching of keywords, mesh terms, and user defined queries. This might include lots of articles that are not of interest since the search is out of context. Those might also miss some important articles because of the exact string matching short comings. Screening the articles manually is a cumbersome and error prone task.

Scholars look for articles where each article presents evidence of the role of one or more proteins in activating a pathway of one or more diseases. For example a search in PubMed and Scopus returns 83,101 articles related to the stroke disease. This magnitude of literature is hard to analyze manually and automated NLP tools that help in the selection of the relevant articles are needed. Enabling the analyses of literature of such magnitude will help health-care providers to build a model for curation and construction of the molecular architecture of pathological and disease conditions. Such model is particularly significant in the context of complex diseases where a large number of proteins contribute to the disease processes such as Alzheimer, Stroke, Myocardial infarct, and Cancer Metastasis.

**This motivates us to use NLP techniques to help extract information of interest from the articles using semi-automatic annotation.**

Existing annotators fall short of performing this task because of the following

1. Most of the existing annotators need a professional software developer to deploy and run them, which is hard for a protein or disease specialist. Knowtator [3] is a protege plugin (protege is a java tool where you make class and define relation) so you have to know how to install and use protege plugins to be able to use Knowtator. Brat [4] is a web server written in python that require a CGI (Common Gateway Interface) capable web server.
2. Some of those annotators could not handle large file sizes such as 100 MB, while the data about Brain Ischemia is a 688 MB.
3. They cannot parse XML and CSV files specific to biomedical indexing databases. XML and CSV file retrieved from online database contain data that might not be of biomedical interest so we need to parse the files before displaying its content.
4. They do not generate needed statistics from the annotated data (such as frequency and co-occurrence of the annotated terms).
5. They are single document based, and cross-document analysis is the target of this work.

Therefore, we developed an annotator that allows for cross-document annotation and surpasses all the above shortcomings. SAMNA is a semi automatic annotator that has the following abilities.

- It is user friendly and easy to install and use.(See appendix A for a tutorial)
- It loads a predefined database of labels with associated terms. and uses it in the annotation process.
- It parses large files and extract important data from files such as titles, abstracts, ids and dates.
- It is able to display documentation aggregated across all articles and files, and allows for inspecting each one alone with its important components.
- It allows the intervention of the specialist to edit and redo the automatic annotations including adding or deleting an annotation and adding or deleting a label.
- It creates statistical metrics that describe the annotations and their occurrences while performing the NLP task.

- It is able to automatically suggest potential annotations within biomedical texts with high accuracy.
- It visualizes the components of articles such as title, abstract, id and date with color sensitive annotations, that highlight occurrences of terms of interest to the NLP task in the text.
- It takes expert rules from the specialist that specify features of the target annotations. It uses these rules to annotate additional terms not defined previously in the database. For example  $\forall Pi \in \text{list of protein P}; Pi[1-9][1-9]^*$  is a protein. Protein followed by one or more digit will be considered a new protein. For example if XYZ is a protein in the list, XYZ3 will be annotated as protein.
- It implements a distributional similarity algorithm using DISCO [9] to find new annotations.
- It saves a result file holding the important statistics of annotation occurrences and frequencies for each annotation. In this way SAMNA provides a full annotation spectrum for further analysis by investigators.

SAMNA is a promising tool for medical NLP applications, such as information extraction from medical databases which will help researchers to study cross document relations such as protein to disease relations.

# Chapter 5

## Semi Automatic Annotator

The semi-automatic annotator for medical NLP applications SAMNA, allows the specialist to load files with a specific format such as EXtensible Markup Language (XML), comma-separated values (CSV), and text file TXT. Those files can be extracted from PubMed and Scopus for example. Then SAMNA allows semi automatic annotation of these files.

### 5.1 SAMNA File Extraction

SAMNA loads the files and extracts from them the following needed components upon the request of the user.

- The title,
- The abstract,
- The ID (PMID for example) which is a unique ID that identifies the publication, and
- The date which is the date of publication

In case scholars were interested in more details such as the results or the figures of an article, SAMNA is also supported with scripts that can extract other parts of the article if needed and if available. Typically the title and abstract are enough for annotation because they are manageable and easy for screening. They have the most important information about the publication work and a brief description of the results. Further more lots of papers are expensive and not open access, and lots of paper are not fully digitize yet.

**PubMed XML structure** A search in the PubMed database results in an XML file with the following structures and tags.

In the XML file each article is embodied inside a node named **PubmedArticle** that contains:

- **ArticleTitle** is the node holding the title of an article.
- **Abstract** is the node enclosing the abstract of an article.
- **PMID** is the node containing the PMID of the article.
- **DateCompleted** is the node having the date of the article.

Several other details exist in the XML file that SAMNA ignores for now such as journal details, authors names, and comments.

**Scopus CSV and TXT Structure** CSV file are organized by column names. Each row represents an article. SAMNA is interested in

- The title column.
- The abstract column.
- The id column.
- The publication year column.

Several details exist in the CSV file that SAMNA ignores for now such as Authors names column, article link column, and source column.

SAMNA is a user friendly annotator tool that have a navigation process, caching process, and automatic annotation algorithm

## 5.2 SAMNA Navigation

SAMNA have a user friendly navigation process.

- SAMNA allows the specialist to go between extracted articles one by one, forward and backward. It offers the choice to do that by clicking the next and previous button or through shortcut keyboard navigation keys.
- SAMNA applies the exper rules and the distributional similarity annotation algorithm for each article and allows the user to validate the automatically found annotations.
- SAMNA allows the specialist to go to a specific article number. This gives the specialist the ability to continue working after a break.



- SAMNA allows the specialist to go through and annotate all abstracts automatically while extracting statistics about annotation occurrences and frequencies.
- SAMNA implement shortcut keyboard for every important action. Thus the specialist can annotate the articles in a faster way. For example the specialist can start highlighting the article while reading, and when he finds a term that should be annotated he click CTRL + (the number of word he want to annotate). SAMNA will annotate the number of words he specified starting with the last highlighted word.

### 5.3 SAMNA Data Model

SAMNA has a caching system to load already defined list of annotation labels and associated terms. SAMNA stores new annotations and labels, and saves statistics using spreadsheet files.

- SAMNA can work on several projects each project is organized in a directory named a work space. SAMNA prompt the user for the work space at first. The work space may contain the annotation label and term lists.
- SAMNA saves the annotations in a result database. The specialist can have some statistics about each annotation and its frequency of occurrence. Each row in the result file is an annotation. The row contain annotation term, the id of the article, the label it belong too, the origin term from which the annotated term is deduced, the date of the article, the section where the annotation was found, and the position of the annotation in text.
- SAMNA saves annotation labels and terms in a spreadsheet format. Those spreadsheets are used in the annotation process. It also saves editing action in a spreadsheet format so the action will be memorized.

### 5.4 SAMNA Annotation

SAMNA has a direct string matching annotation process, and an automatic process that suggests new annotation.

- SAMNA annotates words according to some already defined databases of labels. Label databases can be filled manually by a professional person or can be constructed automatically by the tool. This provides the ability of adding a term to the database of labels. This databases contain terms that should be annotated and visualized with a specific legends if found in the document. For example label can be protein, terms can be the proteins names and legend can be a color highlighter.

- SAMNA highlights exact matching words and highlights term matches according to predefined expert rules.
- SAMNA applies predefined specialist rules to catch new annotation.
- SAMNA applies a distributional similarity algorithm to find and suggest new annotations.

## 5.5 SAMNA User Editing

SAMNA allows the user to edit annotation terms and label, and to exclude a term or an article from the annotation process.

- SAMNA enables the specialist to add or remove a label. To add a label the specialist has to choose the label name and the associated legend. When the specialist adds a new label, a database of this label will be created and the specialist will be able to add and remove annotations to the label database.
- SAMNA enables the specialist to add and remove new annotations to the label database. when adding (or removing) an annotation the annotation will be highlighted (or unhighlighted) in all other article. any addition (or deletion) of an annotation will affect the label database.
- SAMNA enables the specialist to exclude a term from annotations in a specific article.
- SAMNA enables the specialist to exclude a article from annotations. This article will not be shown in results.
- SAMNA allows the specialist to choose whether to apply the rules while searching for annotations in text.
- SAMNA gives the ability to undo actions.
- SAMNA offers shortcuts to do the important actions mentioned above.

## 5.6 SAMNA Analysis

SAMNA provides the results of the annotation in a spreadsheet files to be used for the analysis such as:

- Inter-annotation agreement.
- Graph analysis from detected entities.
- Cross document analysis with other databases.

All these features are shown in the java GUI buttons, right click menu, and keyboard shortcuts which made SAMNA a user friendly tool. Details about the GUI and the buttons places and functions can be find in the tutorial guide, Appendix. A.

# Chapter 6

## Method

SAMNA uses three different methods to annotate a word in the text. It uses a string matching algorithm where it annotates words that match the predefined terms. It also uses a Knowledge base algorithm to annotate words that syntactically look like predefined regular expression rules associated with terms and labels. Finally, it uses a statistical algorithm based on distributional similarity to suggest annotations.

### 6.1 Direct String Matching

SAMNA loads the predefined database of terms in a lookup has table. It reads the text word by word and checks whether the word exists in the lookup table. If a match is found, SAMNA creates an annotation with the associated label for the match, highlight it in the interface , and saves it in the result spread sheet file. For multi word terms, SAMNA uses the first word of the term as a hash key and saves the rest of the term in the has entry for further matching (Figure 1).

From 7,403 predefined protein names we were able to find 9,053 exact matches in 36,818 abstracts for spinal cord injury XML file. Those 9,053 exact matches are for 915 unique protein terms.

### 6.2 knowledge Base Rules

SAMNA takes a set of regular expression associated with a set of terms. The regular expressions form a knowledge base of rules. The regular expressions are written with the help of the expert scholar; i.e the protein expert in the stroke and spinal cord studies. Those following rules are examples of rules implemented to capture proteins names

- Allow a number after a word;  $\exists P_i \in \text{list of protein P}; P_i[1-9][1-9]^*$  is a protein.

```

1 public void SAMNStringMatching (text , labels)
2 {
3 h= CreatHashTable (labels) ; // see figure 2
4 f o r each word w in t ext
5 {
6 r= lookup (h ,w) ;
7 if (r is multiword )
8 {
9 if (Check (r.rest , Text (After(Pos(w))))))
10 {
11 annotate (w +Text(After(Pos(w))) ) ;
12 }
13 }
14 else if ( r is true )
15 {
16 annotate (w,r.label) ;
17 }
18 }
19 }

```

Figure 6.1: String matching algorithm

```

1 public HashTable CreatHashTable (labels)
2 {
3 HashTable h = new HashTable() ;
4 for each term t in labels
5 {
6 if (t is one word)
7 {
8 h.Add(t ,true) ;
9 }
10 else
11 {
12 h.Add( firstWord(t) , RestWords(t));
13 }
14 }
15 return h;
16 }

```

Figure 6.2: Create hash table from label spread sheet

- Add and remove 's' for plural;  $\exists P_i \in$  list of protein P;  $P_i$ "s" is a protein and  $\exists P_i$ "s"  $\in$  list of protein P;  $P_i$  is a protein.
- Add an asterisk "\*" to a word;  $\exists P_i \in$  list of protein P;  $P_i$ +"\*" + [A-Z][A-Z]\* is a protein.
- Add a dash "-" to a word;  $\exists P_i \in$  list of protein P;  $P_i$ +"-" + [A-Z][A-Z]\* is a protein.
- Add a "P-" and "C-" prefix to a word;  $\exists P_i \in$  list of protein P; "P-" +  $P_i$  and "C-" +  $P_i$  is a protein.
- Consider word with parenthesis expressions following them '(' or ')';  $\forall P_i \in$  list of protein P; " $P_i$ (" + "\*" + ")" is a protein.
- Consider words with slash and backslash '/' or '\ ' without them;  $\exists P_i \in$  list of protein P;  $P_i$ +'/' + "\*" is a protein.
- Similar rules were also included to treat special characters such as commas, points, and brackets.

The above rules were hard coded to optimize their run time. SAMNA also accepts user defined regular expressions. Using knowledge expert rules, we were able to find 10,504 protein names in 36,818 abstracts for spinal cord injury XML file. Those 10,504 protein names were from 1,458 protein names deduced from the same 915 unique protein names. This method increases our protein finding from 915 till 1458

## 6.3 Distributional Similarity Algorithm

SAMNA uses an automatic statistical algorithm based on distributional similarity to boost annotation effectiveness. The distributional similarity algorithm is illustrated in Listing 5.3, 5.4, and 5.5. The distributional similarity algorithm works as follows.

1. It computes  $S_i$  and  $C_i$  which are the the set of the top similar and collocated words of each term  $t_i$  in the terms using Disco [9] tool. The similarity and collocation scores  $R_{Sij}$  and  $R_{Cij}$  are calculated based on a pre computed corpora index. We used the PubMed index for the stroke and spinal cord. PubMed data packet is a packet provided by disco containing the index of approx. 100,000 medical articles from the PubMed Open Access database (July 2007).
2. The top similar and collocated words are selected based on a user defined threshold  $T_S$  and  $T_C$  (Figure 3 and 4).

```

1 Public HashTable GetOneWordSimilarity (oneWordTerm, disco)
2 {
3   HashTable hSimilarWordsAndScores = new HashTable();
4   SimilarWordsAndScores= disco.SimilarWordsAndScores (
      oneWordTerm) ;
5   for each word sw in SimilarWordsAndScores
6   {
7     if (sw.score>TS)
8     {
9       hSimilarWordsAndScores.Add(sw.word, sw.score);
10    }
11  }
12  return hSimilarWordsAndScores ;
13 }

```

Figure 6.3: Get similarity for 1 word protein

```

1 Public HashTable GetOneWordCollocation (oneWordTerm, disco)
2 {
3   HashTable hCollocationWordsAndScores = new HashTable();
4   CollocationWordsAndScores= disco.CollocationWordsAndScores (
      oneWordTerm) ;
5   for each word cw in CollocationWordsAndScores
6   {
7     if (cw.score>TC)
8     {
9       hCollocationWordsAndScores.Add(cw.word, cw.score);
10    }
11  }
12  return hCollocationWordsAndScores ;
13 }

```

Figure 6.4: Get collocation for 1 word protein

- It uses an algorithm to compute the similarity and collocation of multi-word terms, which is not directly supported by DISCO. This algorithm divides the multi-word term into single words and computes the similarity and collocation of every word. then it computes the intersection of the similarity of all the words. It associates each word in the intersection set with its minimum score across all words. It does the same for the collocation sets (Figure 5).

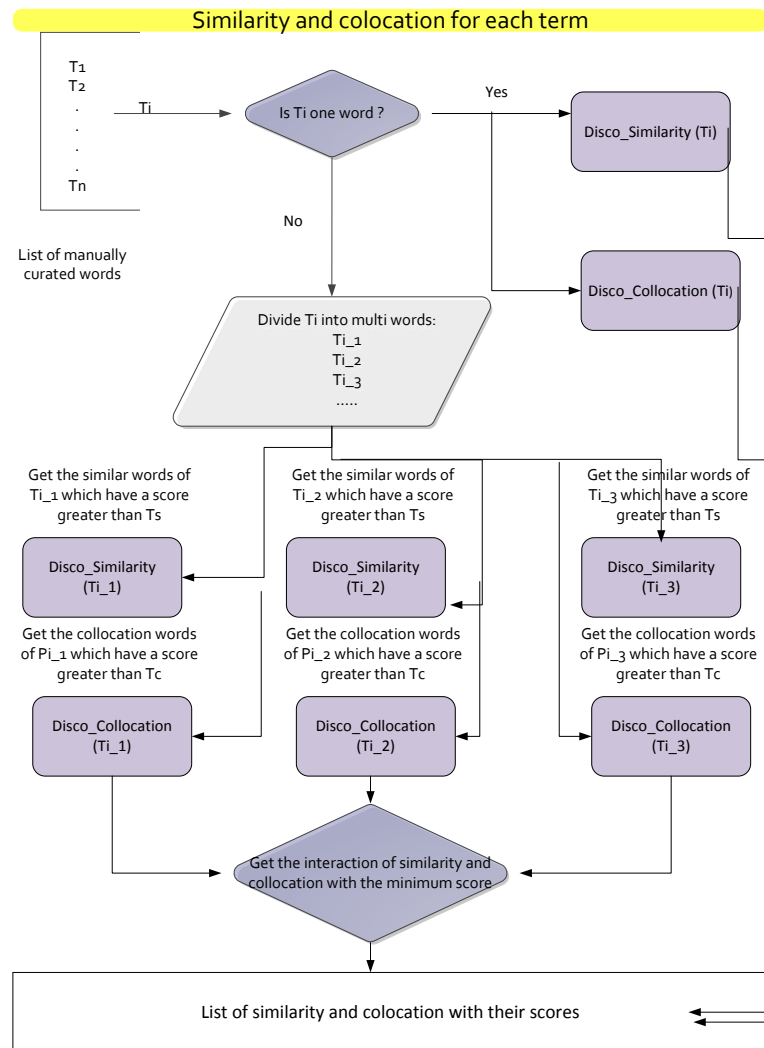


Figure 6.5: compute similarity and collocation between all words in text

- It computes the total similarity and collocation score for every suggested terms. After computing the similarity and collocation of each term  $S_i$  and  $C_i$ , a lot of intersection between similar and collocated words should be



found. To compute the score of each similar and collocated word we use the equations:

- (a) Let  $R_{sij}$  be the similarity score of  $S_{ij}$  which is a similar word to the term  $T_i$ .
  - (b) If  $S_{fj}$  which is a similar word to the term  $T_f$  is equal to  $S_{ij}$  having a similarity score  $R_{sfj}$ .
  - (c) Do increment the number of repeated time for the word  $S_{fj}$ .
  - (d) Update the score of suggested similarity word ( $S_{fj}=S_{ij}$ ).  $S_{fj} = R_{sij} + ((R_{sfj} * (\delta_1 + \text{repeated times} * \delta_2))$ .  
 $\delta_1$  and  $\delta_2$  are constant factor to increment the similarity and collocation score
5. It Takes the suggested similarity words that have a total score above a user defined threshold **to have the one word similarity list**. (Figure 6 and 7).
  6. It uses the collocation of those similarity words to compute the multi word suggestion.(Figure 8)

After running this algorithm on the spinal cord injury file, provided to the algorithm an input of 7403 protein name (same protein names as above). The algorithm was able to suggest 8605 protein names with 530 new unique protein names deduced from 86 distinct suggested protein names with a recall of 81.12

## 6.4 Choosing $\delta_1$ and $\delta_2$

To choose the best value of  $\delta_1$  and  $\delta_2$  we did alternate the value of  $\delta_1$  and  $\delta_2$  computing the precision of the 500 highest score suggested words. Table 1 and Figure 9 represent our findings which indicate that we should set  $\delta_1 = 1000$  and  $\delta_2 = 10$  to get a precision of 89.1% for the first 500 words.

```

1 Public SimilarityListGetOneWORDSimilarityList(Terms, disco)
2 {
3   int delta=1000;
4   int delta2=10;
5   globalsimilarityscorethreshold=68445871;
6   foreach Termt in Terms
7   {
8     St=GetOneWordSimilarity(t, disco);
9     foreach word $S_ti$ in $S_t$
10    {
11      if(similarityScores.containsKey(Sti.word))
12      {
13        similarityScores.put($S_ti$.word(), similarityScores.get($S_ti$.
14          .word)+($S_ti$.score*delta1+(similarityRepeadted.get(entry.
15            getKey()*delta2)));
16        similarityRepeadted.put($S_ti$.word, similarityRepeadted.get(
17          $S_ti$.word)+1);
18      }
19    }
20  }
21 }
22 foreach entry in similarityScores.entrySet()
23 {
24   if(entry.getValue().>globalsimilarityscorethreshold)
25   {
26     oneWordSuggestions.add(entry.getKey());
27   }
28 }
29 return oneWordSuggestions;
30 }

```

Figure 6.6: Get the 1 word suggested similarity List

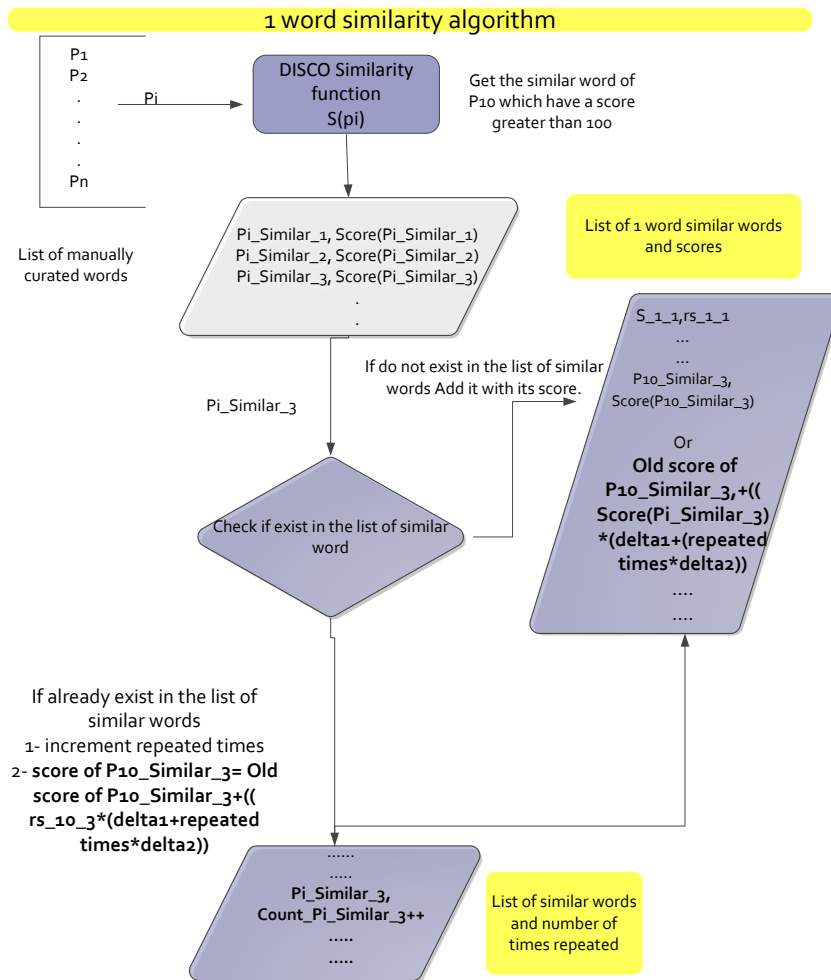


Figure 6.7: distributional similarity algorithm

```

1 Public List GetTwoWORDSsimilarityList(Terms, disco)
2 {
3 oneWordSimilaritySuggestions=GetOneWORDSsimilarityList(Proteins
4     , disco);
5 int delta=1000;
6 int delta2=10;
7 globalcollocationscorethreshold=1435699;
8 foreach similarword sw in oneWordSimilaritySuggestions
9 {
10 collocations=GetOneWordCollocation(sw, disco);
11 foreach collocation c in collocations
12 {
13 if ( collocationScores.containsKey(c.word))
14 {
15 collocationScores.put(c.word, collocationScores.get(c.word)+(c.
16     score*delta1+(collocationRepeadted.get(c.word)*delta2));
17 collocationRepeadted.put(c.word, collocationRepeadted.get(c.word)
18     +1);
19 }
20 else
21 {
22 collocationScores.put(c.word, c.score);
23 collocationRepeadted.put(c.word, 1);
24 }
25 }
26 foreach entry in collocationScores.entrySet()
27 {
28 if (entry.getValue()>globalcollocationscorethreshold)
29 {
30 oneWordCollocationSuggestions.add(entry.getKey());
31 }
32 }
33 for (inti=0;i<oneWordCollocationSuggestions.size();i++)
34 {
35 for (intj=0;j<oneWordSimilaritySuggestions.size();j++)
36 {
37 multiWordSuggestions.add(oneWordSimilaritySuggestions.get(j)+"
38     "+oneWordCollocationSuggestions.get(i));
39 }
40 }
41 //for three words suggestion we add the collocation for the
42     collocations words
43 return multiWordSuggestions;
44 }

```

Figure 6.8: Get the multi word similarity List

alternating $\delta_1$ and $\delta_2$		
$\delta_1$	$\delta_2$	Precision
1	0	0.879732739
10	0	0.879732739
100	0	0.879732739
1000	0	0.879732739
1000	5	0.884955752
1000	10	0.89135255
1000	12.5	0.887417219
1000	20	0.889867841
1000	100	0.887417219
5000	1000	0.887168142
1000	1000	0.885462555

Table 6.1: precision while alternating  $\delta_1$  and  $\delta_2$

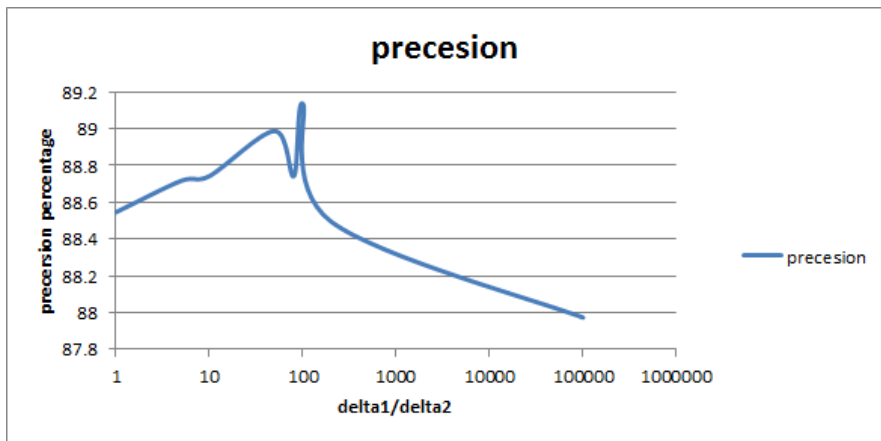


Figure 6.9: Precision While alternating  $\delta_1$  and  $\delta_2$

# Chapter 7

## Implementation

SAMNA is a java tool, it is composed of three packages: file reader, highlighter, and annotator.

### 7.1 File Reader

The file reader package is the package responsible of parsing the XML, CSV, and TXT files containing the abstracts and the needed information. Reading from spread sheet files, writing to them, and deleting from them. Spread sheet files represent the database structure of SAMNA.

The files reader package consists of the following.

1. XmlParser class responsible of parsing the XML file taken from the PubMed database it has two public methods:
  - (a) GetXmlAbstracts:
    - Takes XmlFilePath as String which is the path of the XML file,
    - Returns an array list of hash map of strings in each hash map there are: title, abstract, PMID, and date.
    - Create a serialized object of this array list of hash map of strings in the same file directory, so the next time it will take much less time to load the abstract using the second method GetXmlAbstractsSerialized.
  - (b) GetXmlAbstractsSerialized:
    - takes SerializedFilePath as String which is the path of the serialized object file
    - Returns an array list of hash map of strings in each hash map there are: title, abstract, PMID, and date.

XML parser parse XML files using java Document Object Model (DOM) parser, XML parser searches for nodes named <PubmedArticle>, which represents an article and extract from it the nodes containing title, abstract, PMID, and date which are respectively <ArticleTitle>, <Abstract>, <PMID>, and <DateCompleted>.

2. TxtReader class responsible of parsing CSV and txt file taken from scopus database. in this class an additional library is used, which is opencsv-2.3.jar to easily read csv files. it has two public method each for one file type
  - (a) GetTxtAbstracts:
    - It takes TxtFilePath as String which is the path of the txt file.
    - Returns an array list of hasp map of strings in each hasp map there are: title, abstract, PMID, and date.
  - (b) GetTxtAbstractsFromCSV:
    - it takes CSVFilePath as String which is the path of the CSV file
    - Returns an array list of hasp map of strings in each hasp map there are: title, abstract, PMID, and date.

The CSV library allows to easy parse the csv file, by choosing the column number and assign its content.

- PMID column 18.
  - Date column 2.
  - Title column 1.
  - Abstract column 14.
3. ExcelReader class, because our databases are now in excel format, we use the excel reader class that has 11 public methods used to
    - (a) Get predefined terms from labels databases along with the suggested terms.
      - Method name: GetAllProteinAndSimilarWords.
      - Input: a string, which is the path of the excel file of labels.
      - Output Array List of Multi map of String and Array List of strings. first multi map are the predefined database of terms, second multi map is the one word suggest terms, third multi map is the two word suggested terms, and the fourth multi map is the three words suggested terms.
        - The first string is the first word of the term.
        - the array list of strings are the rest of words constructing the term.

- Multi map is an implementation of hash map made by google.
  - this hash table implementation made the query toward the label names very fast, because we have to search for each word inside more than 8000 label names, so without this hashing implementation the process will be incredibly slow.
- (b) Read Labels names.
- Method name: GetAllLables.
  - Input: a string, which is the path of the excel file of labels names.
  - Output: Array list of strings which are the labels names excluding the default protein label.
- (c) add a new label in addition to the default label.
- Method name: AddLableToList.
  - Inputs: 3 strings: the path of the excel file of labels names,
    - the new label name, and
    - the new label color.
  - Output: void.
- (d) add a label to its database
- Method name: AddProteinToList.
  - Inputs: 2 strings: the path of the excel file of label, containing its terms, and
    - the new term.
  - Output: void.
- (e) open the excel file
- Method name: open.
  - Input: a string which is the path of the excel file
  - Output: void.
- (f) close the excel file
- Method name: close.
  - Input: a string which is the path of the excel file
  - Output: void.
- (g) delete an annotation from its database
- Method name: RemoveProteinFromDatabase.
  - Input: 2 strings: the path of the excel file, and
    - the name of terms to be deleted.
  - Output: void.
- (h) not adding a word to the result file if it is find in a specific abstract,



- Method name: findRow.
  - Input: 4 strings: the path of the excel file,
    - PMID of the article where the word should not be considered for annotation,
    - the date of the article where the word should not be considered for annotation, and
    - the term that should not be considered in the specific article.
  - Output: an integer which is the row number where this term is founded.
- (i) add annotation to result file
- Method name: AddProteinRecordToResults.
  - Input: 4 strings: the PMID of the article where the annotation was founded,
    - the date of the article where the annotation was founded,
    - the term that should be added to the result file, and
    - the term to which this annotation belongs
  - Output: void.
- (j) add a term to not be considered in a special article
- Method name: NotProteinRecord.
  - Input: 5 strings: the path of the excel file,
    - the PMID of the article where the term was founded,
    - the date of the article where the term was founded,
    - term that should not be considered if found in this specific article, and
    - the label to which this term belongs.
  - Output: void.
- (k) remove annotation from result file
- Method name: RemoveProteinFromResults.
  - Input: 2 strings: the path of the excel file, and
    - annotation that should be removed from result file.
  - Output: void.

In this ExcelReader class we used

- POI library (poi-3.10-FINAL-20140208.jar) to manipulate excel file. POI is a library made by Apache to manipulate Microsoft office Documents. [17].

- commons lang3 library for string manipulating (commons-lang3-3.3.2.jar), common lang library is made by Apache.
- Google library named (guava-16.0.1.jar) to use a MultiMap object which is an implementation of HashMap.[18].

Note that we are opening the excel result file one time at the start of the tool and close it one time at the close of the tool, instead of opening and closing the file each time we want to write on it. This trick save us a lot of time.

## 7.2 Highlighter

The highlighter package has only one class which is the HighlightPainter, HighlightPainter is responsible of highlighting and removing the highlight from the document, in this class we don't need any additional library except for the one mentioned above.

HighlightPainter class has 4 public methods

1. highlight\_version2: it is the mainly use highlight function it takes as parameters:
  - (a) PMID: the id of the article we are highlighting.
  - (b) Date: the Date of the article we are highlighting.
  - (c) textComp: which is the text area in the main frame.
  - (d) pattern: which is the hash table of the protein terms.
  - (e) myHighlightPainter: the highlighter used.
  - (f) myaddedHighlightPainter: another highlighter used for added protein. The difference between highlighters are basically the color used.
  - (g) Type: the label of the annotation such as protein, antibiotic, or bacteria.
  - (h) SpecialCase: a boolean to tell the function to apply specialist rules or no.
  - (i) workspace: path of the work space.
  - (j) excel: which is an instance of the mentioned excel reader class to be able to access and do changes in the databases.

and return an array list of string containing founded annotations.

Basically highlight\_version2 method :

- takes every word in the abstract and title, apply rules on this words to have a list of words, and search for these words in the hash table of terms, the word will be the key of the hash table.

- If the word is founded it continues to see if the next array list of string which are the value of the key are the next words in the documents.
  - If they are the same so it has founded an annotation.
  - If annotations is founded it highlights them and adds them to the result file, after doing a check on the file containing the annotations that should not be considered in a special article.
2. highlightNonProtein: it highlight non protein with a special color to be noticed when removing a protein from database. it takes as parameters
    - (a) textComp: which is the text area in the main frame.
    - (b) pattern: the non protein to remove highlight from it.
    - (c) myHighlightPainter: the highlighter used as an indicator that this protein is removed.
  3. removeHighlights: to remove all highlighter in the main frame, used to clear the text area. removeHighlights takes as parameter the text area only.
  4. removeHighlights: to remove highlighter from a specific word, it takes as parameters
    - (a) jTextArea: which is the text area in the main frame.
    - (b) turnLightOff: the non term we want to remove highlight from it.
    - (c) PMID: the id of the article we are highlighting.
    - (d) Date: the Date of the article we are highlighting.
    - (e) Type: the label of the annotation.
    - (f) workspace: path of the work space.

## 7.3 Annotation

Annotation package have two class:

1. the Action class which is a small class with only a constructor made to save the actions done by the users so we can undo them if we want.
2. the most important class which is the SAMNAFrame class which is the main frame of the SAMNA tool. the SAMNAFrame class contain all GUI structure and buttons, in addition to the logic behind these buttons, it contain the ability to do a right click and to do all available function that will be mentioned in section A.  
SAMNAFrame class contain 1 public method which is its constructor, and 18 private methods used by the constructor to :

- (a) Open a file chooser box to choose a file.
  - BrowsLoadCSVAbstract: to choose a CSV file.
  - BrowsLoadTxtAbstract: to choose a txt file.
  - BrowsLoadAbstract: to choose an XML file.
  - BrowsLoadAbstractserialized: to choose a serialized file.
- (b) parse the content of the chosen files and fill the array list of articles from these files
  - LoadCSVAbstract: to load articles from a CSV file
  - LoadTxtAbstract: to load articles from a TXT file
  - LoadAbstract: to load articles from a XML file
  - LoadAbstractserialized: to load articles from a serialized file
- (c) create the right click menu.
  - CreatPopupMenuOnRightClick: method responsible of creation of the right click menu, and the function of each option.
  - RenderPopupMenu: method used to add an option to the right click menu. RenderPopupMenu method is used when we add a new label, so an option to add an annotation to this new label is added to the right click menu.
- (d) allow the intervention of specialist to add or remove annotation and labels.
  - AddToProteinDatabase: used to add an annotation to its label database.
  - AddLableToDatabase: used to add a new label to the labels database.
  - RemoveProteinFromDatabase: used to remove a term from its label database.
  - RemoveProteinFromResult: used to remove an annotation from the result file
  - RemoveProtein: used to remove default annotation.
  - AddProtein: used to add default annotation.
- (e) highlight the articles
  - RenderFrame: main method used to highlight the main frame.
  - AddToHighlightedWordsSection: method to add annotations to the list on the right side of the frame.

The constructor method also contain:

- The structure and design of the GUI.
- The shortcuts creations.

# Chapter 8

## Experimental results

We provided SAMNA to medical scholars and worked with them on annotating text corpora for two diseases: 1) stroke and 2) spinal cord injury. we discuss the two case studies below.

### 8.1 Stroke Case Study

The burden of ischemic stroke is still the highest among all neurological diseases despite tremendous efforts devoted to prevention, management, treatment and rehabilitation of stroke patients [19], [20]. After decades of research and hundreds of clinical trials on ischemic stroke, the full spectrum of pathophysiological processes has not yet been elucidated neither has a final and terminal treatment been devoted. Preclinical and clinical studies have predicted that a single-action-single-target paradigm is not the optimal approach to treat stroke and that multi-action-multi-target paradigms are required [21]. Eventually, there is a sincere need to compile efforts to understand the evolution of different mechanisms after ischemic stroke and the relation of this to disease outcome and possible interventions. We used SAMNA coupled to systems biology and network analysis tools to analyze the complex protein interaction network that occurs after stroke. To do that, we curated and annotated a brain-ischemia metaproteome (BIMP) through literature mining and applied graph theory algorithms to analyze and dissect the corresponding brain ischemia interactome (BII). The analysis uncovered a rich-club organization in the BII and provided insight into the mechanisms predominating early and subsequent phases of ischemic stroke. In addition, we used network analysis in the context of drug-protein interaction that showed that estrogen is still the most interesting intervention in ischemic stroke.

### 8.1.1 Method

**Extraction of Target Dataset** An overview of the research paradigm of this study is shown in Figure 1. Literature on brain ischemia was extracted from PubMed using the MeSH term (Brain Ischemia) and the search key (Stroke OR brain infarct\* OR cerebral infarct\* OR brain ischem\* OR cerebral ischem\* OR ischemic brain injury) as well as from references of reviews and extracted papers. Abstracts were screened for relevance by two investigators using the title. Selected abstracts were then processed into SAMNA for extraction of protein entities that exhibit association with ischemic stroke. A detailed description of the selection process is illustrated in Figure 3.

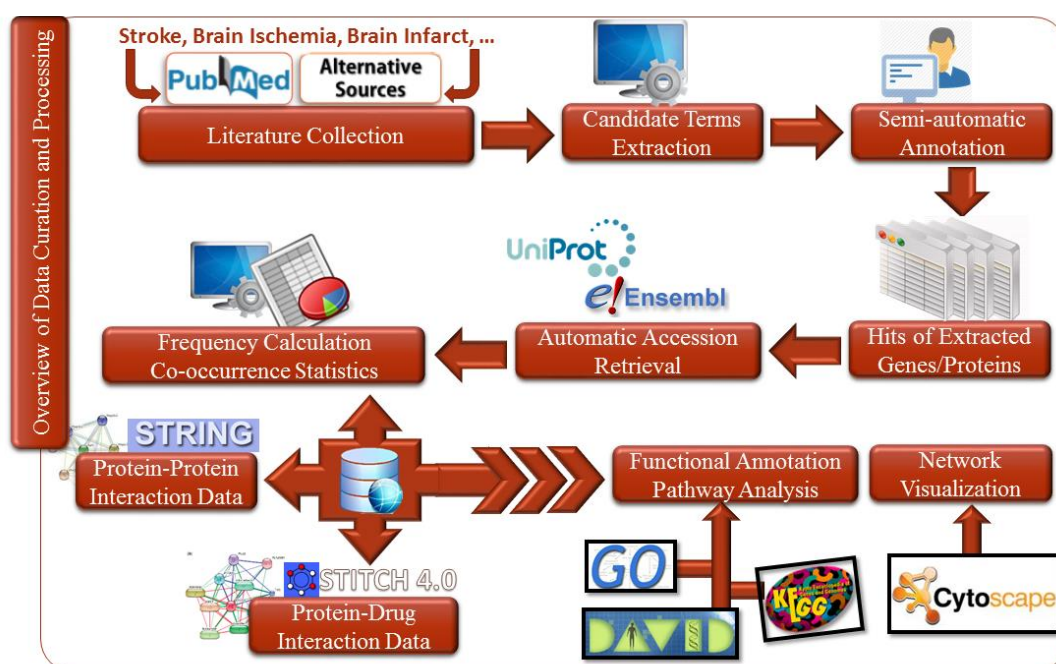


Figure 8.1: overview of data curation and processing

**Text Annotation and Accession Mapping** Different versions of gene and protein names were exported from UniProt (<http://www.uniprot.org>) and HUGO (<http://www.genenames.org>) databases. Abstracts were first matched with the databases for similar terms to be annotated. Exact matches and close matches are annotated as terms of interest and extracted into a separate dataset each identified by the papers ID. A human annotator with experience in both stroke and proteomics verifies the captured term and skims the abstracts using SAMNA for additional terms. With each new term captured, the original database is updated for later use. Captured terms were then mapped automatically using a C# tool that communicates with UniProt and HUGO databases for retrieving human orthologs of captured entities and the corresponding accessions. Frequency of each

accession was calculated as the number of distinct reports with the studied accession. Figure 2 illustrates the details of the text annotation process leading to the captured set of accessions.

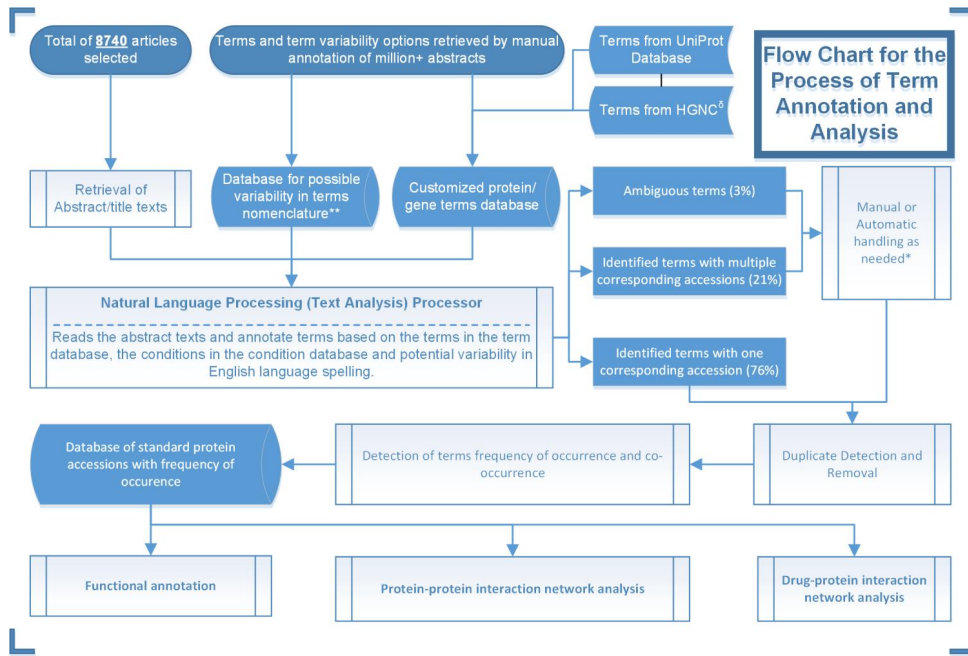


Figure 8.2: captured set of accessions

**Functional Annotation** The list of accessions retrieved from literature was functionally annotated using DAVID [22] for GO cellular component, GO biological processes, KEGG pathways, tissue expression and Genetic Association of Diseases database. Protein-protein interaction (PPI) data for the BIMF was retrieved through STRING database that includes curated PPI data from several other databases [23]. PPI data obtained from STRING is then mapped into the full Brain Ischemia Interactome using Cytoscape 3.1.1 [24]. In addition, data on protein-drug interaction was retrieved from GeneCodis and STITCH. GeneCodis allows for enrichment analysis of gene-drug interactions within a network using PharmGKB database11 while STITCH allows for analysis of chemical-gene interactions within a network using a dataset of 3 million chemical agents [25]. Drug or chemical targets are defined in our study as those proteins and genes with which the drug or chemical interacts and affects regulation.

**Interactome Graph Analysis** Graph Measures: Examination of the topology of the BII network using graph theory was performed through Systems Biology and Evolution MATLAB Toolbox (SBEToolbox) and Cytoscape [25], [26].

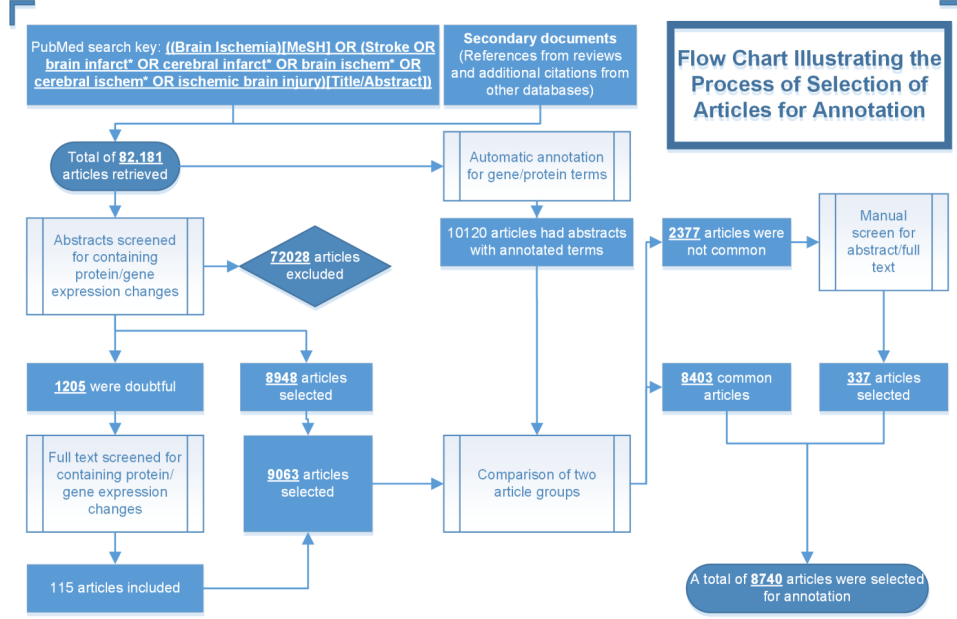


Figure 8.3: articles selection process

Characteristic measures of network organization were computed including node-specific degree  $k$ , clustering coefficient, path length, betweenness centrality, and modularity. Power-law degree distributions and adjacency matrices of the networks were generated in MATLAB (Mathworks, R2013a).

**Rich-Club Analysis** The emphasis of this work is the detection of a rich-club organization among the nodes within the network of the BII. A rich club is a set of high-degree nodes that are more densely interconnected than predicted by the node degrees alone [27]. A rich-club coefficient  $\varphi(k)$  is computed over the range of degrees in the network as previously described by Colizza et al [27]. For a given degree distribution  $k_1, \dots, k_n$ , rich-club coefficient for each degree  $k$  is calculated as the number of edges among nodes with degrees higher than  $k$  divided by the maximum possible number of edges among those nodes (Equation (1)).

$$\varphi(k) = \frac{2E_{>k}}{N_{>k}(N_{>k} - 1)} \quad (1)$$

where  $N_{>k}$  is the number of nodes with a degree higher than  $k$ , and  $E_{>k}$  is the number of edges among those nodes.

To calculate normalized rich-club coefficient, we generated 10,000 random networks with the same degree distribution as the network of interest as described by Viger and Latapy [28]. The average of rich-club coefficients of the random networks  $\varphi_{random}(k)$  is calculated, and the normalized rich club is computed as



in Equation (2):

$$\varpi(k) = \frac{\varphi(k)}{\varphi_{random}(k)}(2)$$

The normalized rich-club coefficient was calculated from the lowest degree to the second highest degree encountered in the BII. A normalized rich-club coefficient  $\varpi(k) > 1$  indicates the presence of a rich-club organization in the network.

## 8.1.2 Results

**Curation and Annotation of First Brain Ischemia Meta-proteome** To extract our target Brain Ischemia Meta-proteome (BIMP), we retrieved 82,181 articles through PubMed and other databases using a combination of MeSH terms and sensitive search keys (Figure 3).

We performed semiautomatic annotation of proteins reported to be involved in stroke pathogenesis and recovery. A combination of preset protein nomenclature database and manual validation was used to ensure sensitive and specific capture (Figure 2). A total of 927 curated proteins were used for future analyses after mapping proteins from different species to human orthologs. As predicted, tissue-plasminogen activator (t-PA) was the most frequently reported protein.

Functional annotation and clustering of proteins in the BIMP were performed

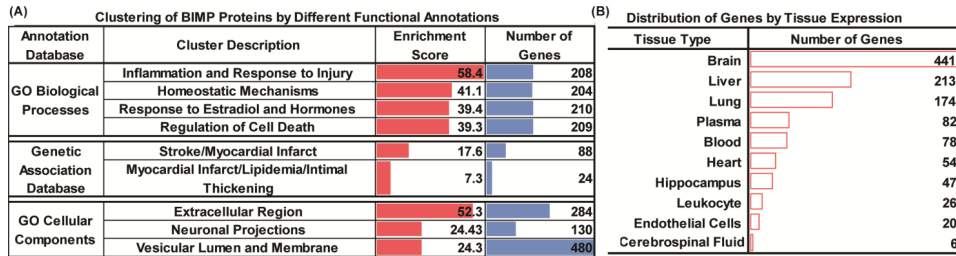


Figure 8.4: clustering of proteins in the BIMP expressed by brain tissue

using DAVID for enriched GO (Gene Ontology) biological processes, cellular components and tissue expressions [29]. As summarized in Figure 4, BIMP proteins are predominantly expressed by brain tissue (Figure 4B), cluster by GO cellular component to extracellular space, neuronal projections and vesicular clusters, and are primarily enriched for inflammatory processes (Figure 4A). Furthermore, KEGG 7 pathway annotation reveals that complement and coagulation cascade (CCC) is the most enriched pathway followed by calcium signaling and mitogen-activated-kinase (MAPK) pathways (Figure 5). Notably, The CCC pathway appears to have minor intersection in terms of common components with other pathways compared to the other major pathways.

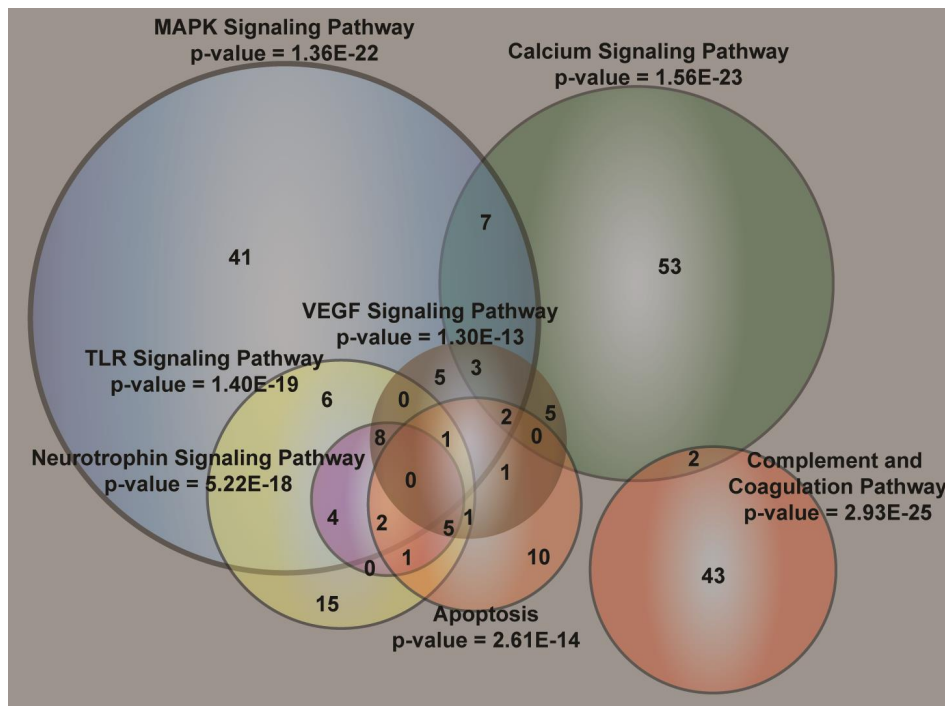


Figure 8.5: Venn Diagram of the distribution of BIMP proteins on different significantly enriched KEGG pathways

**Construction and Analysis of Brain-Ischemia Interactome** Protein-protein interactions among components of the BIMP were retrieved via STRING database [11] and the network representing interactions within the BIMP was analyzed through Cytoscape [30] and MATLAB (R2013a, The MathWorks, Inc.). A full visualization of the full network is shown in Figure 6 clearly indicating the need to use automated network analysis tools to reduce the network into an appreciable form.

Markov Clustering Algorithm (MCL) was used to identify 16 distinct clusters within the network. Figure 7A shows a reduced form of the interactome given as interactions between major MCL clusters. Functional annotation of enriched GO biological processes in each cluster is summarized in Figure 7B and shows that inflammatory response, regulation of cell death and glutamate receptor signaling are the most influential biological processes within the network of clusters having the highest node-specific degrees. In addition, interconnections were mapped among the three most enriched KEGG pathways in the network showing that components of the CCC pathway are heavily interconnected with proteins in other pathways despite that it has lower intersection in terms of components (Figure 8). This finding is specifically significant for understanding the role of this pathway in ischemic and reperfusion injury.

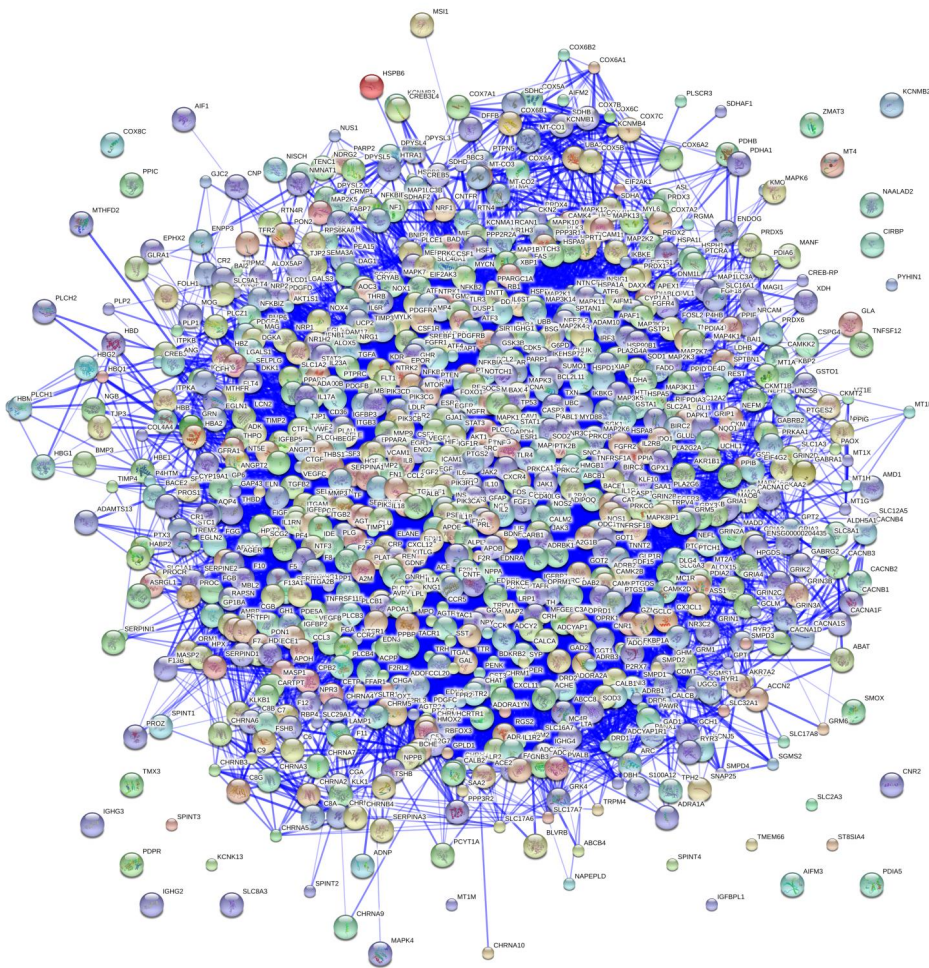


Figure 8.6: Markov clustering of the network of BII

**Rich-Club Organization in Brain Ischemia Interactome** As shown in Figure 9 (A-C), analysis of the brain ischemia interactome (BII) for the existence of rich-club organization has shown the presence of a rich-club for nodes with degrees ranging between 40 and 180 and peaking at degree of 132. Nodes with the highest rich-club coefficient (above 1.3) are highlighted in Figure 9A and defined as the rich-club core. Rich-club analysis was also performed on the three top degree clusters reported in Figure 6 revealing the presence of rich-club organization in clusters 16 and 14 enriched for inflammatory response and regulation of cell death respectively and not in cluster 5 (Figure 7C-7H). Interestingly, comparison of rich-club components to non-rich-club components for frequency of encounter in the curated literature shows that the frequency of rich-club nodes is significantly higher (Figure 9D). Members of the rich-club were found to span multiple pathophysiological pathways that predominantly include inflammatory

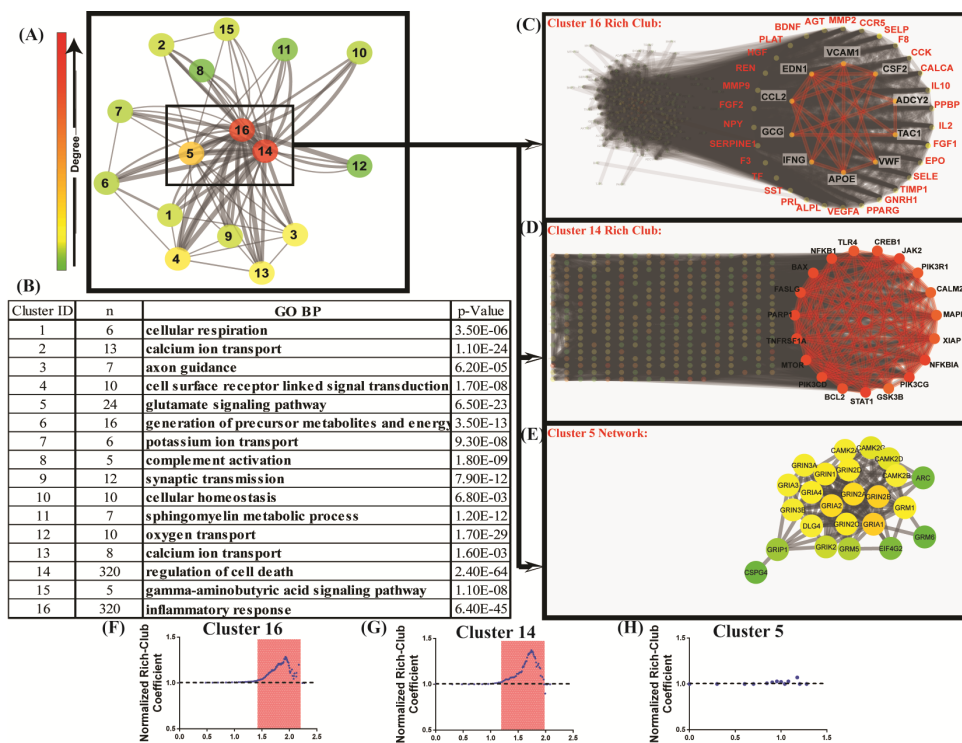


Figure 8.7: Markov clustering of the network of BII

and immunological response mechanisms.

**Estrogen: a Pleiotropic Effect in Stroke Treatment** In the last step, we use findings of network analysis as a screening effect for potential therapeutics. Analysis of protein-drug interactions, performed through STITCH [25] and [31], have revealed estrogen as the most enriched chemical therapeutic within our BII. Targets of estrogen within our BII are shown in Figure 10A and include up to 15% of the nodes in the network. Estrogen was also found to preferentially target nodes within the rich-club (53% of total rich-club components) which was reflected by a significant enrichment on Fischer Exact t-test (Figure. 10C). Eventually, estrogen targets were shown to have significantly higher normalized rich-club coefficient compared to estrogen non-targets (Figure. 10B). Estrogen targets within the BII had significantly higher participation coefficients upon MCL clustering compared to non-targets indicating a more prominent contribution by those nodes to the global network (Figure. 10D). Besides estrogen, other chemical compounds that have similar pleiotropic effect (beneficial or harmful) on targeting components of the BII are reported.

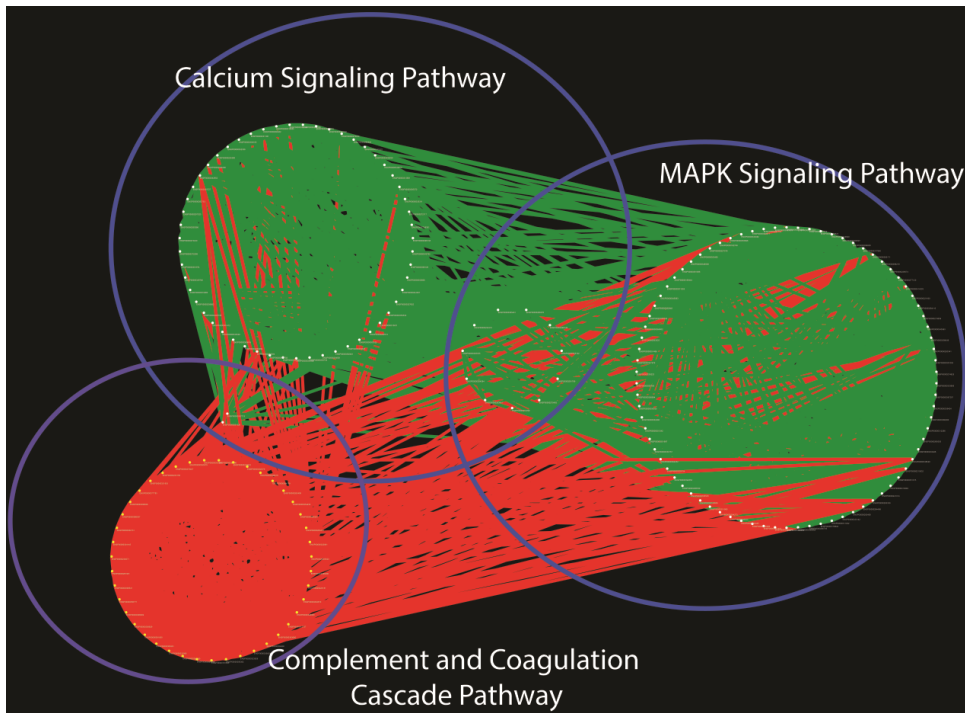


Figure 8.8: Graph of protein-protein interactions among components of major KEGG enriched pathways

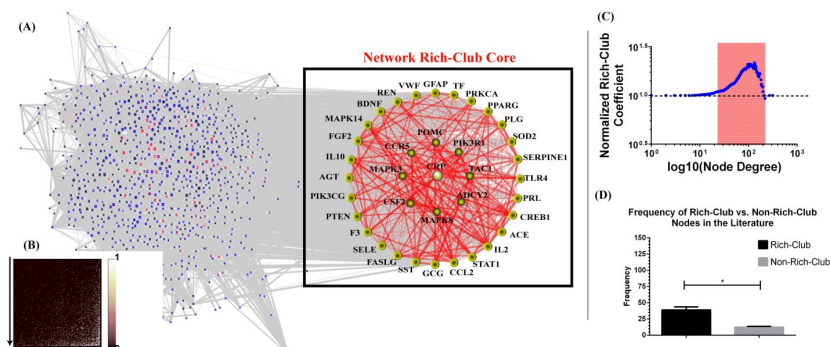


Figure 8.9: Rich-club organization in the BII

### 8.1.3 Conclusion

We used SAMNA to construct, annotate and curate a brain ischemia meta-proteome that reflects the spectrum of protein changes after brain infarct. We used systems biology databases including UniProt [10] for protein accession retrieval and STRING [11] for protein interaction data, to construct the interaction network among the extracted proteins. Curation of the proteome allowed for dis-

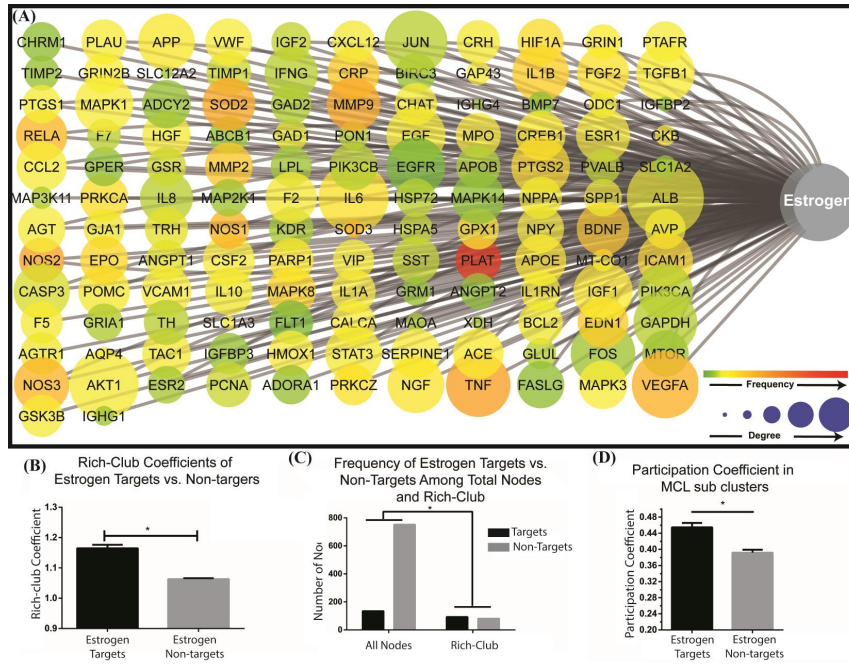


Figure 8.10: Estrogen targets within the BIMP

secting the role of different biochemical pathways in the amplification of injury after stroke and revealed, with the help of graph theory algorithms, the presence of a rich-club organization in the brain ischemia interactome. The detected pathways and the rich-club provide information on the optimal approaches to target injury mechanisms after stroke.

## 8.2 Case study 2 spinal cord injury

Spinal cord injury (SCI) is a prominent cause of disability worldwide, and in the U.S. is second only to stroke as a cause of disability, accounting for 23% of all cases of paralysis [32]. The fact that there is no effective therapeutic intervention for the treatment of SCI highlights the need for a better and more integrative understanding of the molecular mechanisms that promote pathology and determine recovery.

There is limited information available on how pathways involved in these processes connect with each other to result in the pathological outcome. With the aim of providing insight into the interdependency of prominent pathophysiological processes and the molecular disturbances that affect neuronal survival and axonal regeneration, we investigated the full molecular architecture of SCI. In this work, we use SAMNA to extract information from the scientific literature on protein and gene disturbances after SCI. We use systems biology tools and

resources to map interactions among proteins reported to be dysregulated after SCI, and we use network analysis and graph theoretical algorithms to study the network of protein interactions after SCI. We identify hub proteins, critical pathways, and modules involved in SCI pathogenesis, as well as potential therapeutic targets.

### 8.2.1 Materials and Methods

**Extraction of Target Dataset** Literature on spinal cord injury was extracted from PubMed using the MeSH term (Spinal Cord Injuries) and the search key ("Traumatic Spinal Cord"[Title/Abstract] OR "Spinal Cord Trauma"[Title/Abstract]), as well as from references of reviews and extracted papers. After abstracts were screened for relevance by two scientific curators, selected papers were then processed into SAMNA for extraction of protein and gene terms reported in association with SCI.

**Text Annotation and Accession Mapping Tools** To extract protein and gene names and identifiers from the text, we used SAMNA to ensure specificity and sensitivity of capture. SAMNA:

- Extracts different versions of gene and protein names and abbreviations from UniProt (<http://www.uniprot.org>) and HGNC (<http://www.genenames.org>) databases,
- Checks the text of the extracted abstracts with the gene and protein terms extracted from the databases to annotate exact matches and,
- Detects and annotates close matches based on variations of the extracted terms. Variations were defined in the form of computational knowledge base rules to detect differences in spelling, the mentioning of specific and multiple subunits, and variability in the use of special characters and abbreviations

The detected and annotated terms were extracted into separate datasets, each identified by the ID of the source paper. A human annotator with experience in both neurotrauma and proteomics verified the captured terms and inspected the abstracts using SAMNA. SAMNA ran several trials of manual annotation to assess the precision and recall of SAMNA. Each trial included around 1000 annotated terms with their corresponding abstracts. After each trial, the set of rules was updated based on annotators input. Trials were performed with rules updates until SAMNA achieved more than 99.5% on precision and recall measures. Fig. (1a) summarizes the overall approach used for term extraction. Next, an in-house C# program communicated with UniProt and HGNC to retrieve the corresponding accessions of captured terms. The frequency of each accession was computed as the number of distinct reports of accession in question. To avoid

being biased by the literature, there was no assignment of weights for frequency in our network analysis, but terms with a frequency less than three were cross-validated by a second annotator to ensure minimal false positive captures (Fig. 1c).

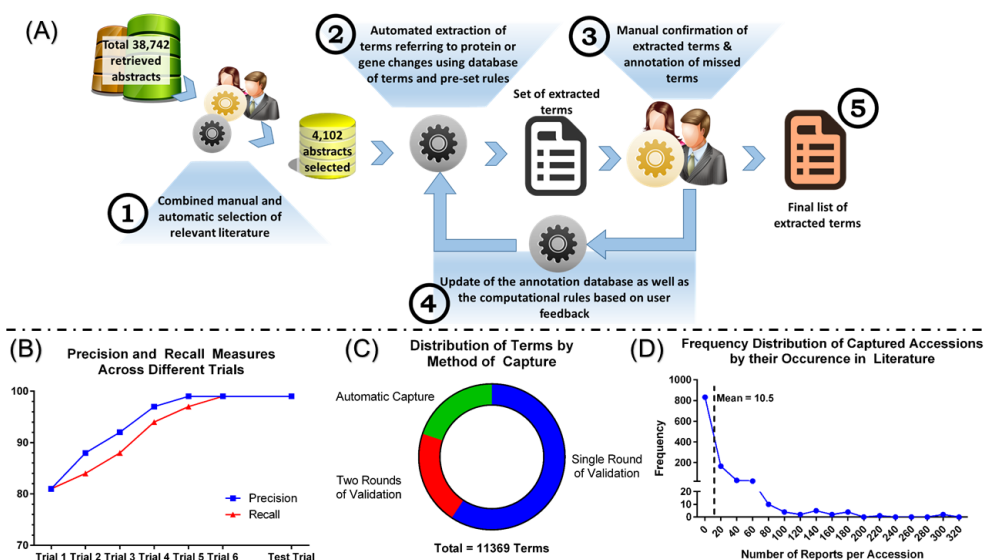


Figure 8.11: SPINAL CORD

**Functional Annotation and Interactome Data** We used the Database for Annotation, Visualization and Integrated Discovery (DAVID) [22] [33] to annotate the retrieved list of accession for GO (Gene Ontology) cellular component, GO biological processes, and KEGG (Kyoto Encyclopaedia of Genes and Genomes) pathways. Functional annotation clustering was also performed through DAVID to identify enrichment scores for different groups of annotations. For clarity of presentation, we report p-values of corrected Fischer exact t-test for enrichment analyses as enrichment scores which are computed as  $\log_{10}(1/p\text{-value})$ . Protein-protein interactions (PPI) were obtained from STRING databases by including interactions among the list of accessions with a combined score higher than 0.4 [11] [34].

**Graph Theory Measures** After construction of the SCI interactome, graph theory was used to examine the topology of the network

## 8.2.2 Results

We retrieved 38,742 abstracts from PubMed searches of which 4,102 abstracts were found to discuss relevant information on protein changes in the context of



SCI. After six trials of manual cross-validation and tool optimization (Fig. 1a), we were able to achieve more than 99.5% on both precision and recall measures, which was further confirmed by a test trial (Fig 1b). We then captured a total of 11,369 terms that included automatically captured and manually verified terms (Fig 1c). Captured terms were mapped to 1,083 unique accessions for proteins or genes associated with SCI pathogenesis. The distribution of those accessions by occurrence in the literature was right skewed with a mean of 10.5 reports per accession (Fig 1d). The full list of curated accessions along with their frequencies and references are shown in the Datasets of all captured terms listed by their UniProt protein accessions and including their frequencies and corresponding references. We named this list an SCI meta-proteome, defined as the protein complement of SCI changes curated from experiments in different animal species.

### **8.2.3 Conclusion**

We used SAMNA to curate the proteome of spinal cord injury (SCI) and we were able to characterize the full molecular architecture of SCI. Analysis of the curated proteome using graph theory revealed a modular organization of disease process that are centered around a core of cell survival decision proteins. Those core proteins provide the best targets for therapy.

# Chapter 9

## Related Work

knowtator is a manual creation annotation tool, knowtator is implemented as a protege plugin, its main strength is in the ability of creating a complex schema using protege class. which is in the same time its weak point, because you have to be a developer to write an annotation schema. schema is a specification of what kind of annotation can be created. knowtator schema can model syntactic and semantic cases in knowtator annotated data can be exported in XML Format, they did implement an inner annotator agreement (IAA) metric, which generate a descriptive report about consistency between annotators. [3].

Brat introduce a web based tool for text annotation, which implement a semantic class disambiguation technique to reduce a 15% of annotation time. brat is supported by NLP technology it aim to be :

- user friendly
- use NLP technique in a wise way.
- support user intervention and judgment ( non technical user such as subject domain expert)
- annotation is aided by an automatic semantic class disambiguation technique.

BRAT it is implemented in python, and need a CGI compatible web server to be used. it is structured for NLP tasks. [4]. Semantic category disambiguation is the algorithm used in BRAT, it is an important sub task of several problem in NLP, and was recently used in assisting other tasks such as annotation. in [35] they introduce a task setting where they return many suggestions for a given annotation, through a beam search with a dynamic width beam.

Wordfreak is a NLP annotation tool, written entirely in java, it is designed to allow components to be added without the need of recompiling the initial source, through a plugin architecture. Wordfreak needs development to create a new annotation schema; it is an automatic tool with some confidence measurement, and annotation choices which help in NLP tasks such as POS and in active learning or correction of automatic annotation. [5].

# Chapter 10

## Conclusion and Future Work

We did introduce SAMNA a Semi-automatic Annotator for Medical NLP Applications. SAMNA help scholars to annotate large corpora with labels and terms of interest. It also uses distributional similarity to discover novel annotations. In addition, it provides the annotating scholar with an intuitive, friendly and efficient interface to navigate and edit the annotations. We did prove SAMNA's need and uniqueness in many medical applications. SAMNA has a promising future in medical NLP application specially in the protein to disease relation. It will always be upgraded technically and functionally to meet the scholars need.

- Upgrade the distributional similarity algorithm to be more efficient in multi-word suggestion as it is in the one word suggestion.
- Integrate the tool with MySQL database as an option instead off using spread sheet files. MySQL can run on many platform. and it is an open source widely use database system.

# Bibliography

- [1] F. A. Zaraket and A. Jaber, “Matar: Morphology-based tagger for arabic,” in *Computer Systems and Applications (AICCSA), 2013 ACS International Conference on*, pp. 1–4, IEEE, 2013.
- [2] C. Nédellec, R. Bossy, J.-D. Kim, J.-j. Kim, T. Ohta, S. Pyysalo, and P. Zweigenbaum, “Overview of bionlp shared task 2013,” in *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 1–7, 2013.
- [3] P. V. Ogren, “Knowtator: A protege plug-in for annotated corpus construction,” in *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume: Demonstrations, NAACL-Demonstrations ’06*, (Stroudsburg, PA, USA), pp. 273–275, Association for Computational Linguistics, 2006.
- [4] P. Stenetorp, S. Pyysalo, G. Topic, T. Ohta, S. Ananiadou, and J. Tsujii, “Brat: a web-based tool for nlp-assisted text annotation,” *EACL 2012*, p. 102, 2012.
- [5] T. Morton and J. LaCivita, “Wordfreak: an open tool for linguistic annotation,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Demonstrations-Volume 4*, pp. 17–18, Association for Computational Linguistics, 2003.
- [6] C. Müller and M. Strube, “Multi-level annotation of linguistic data with MMAX2,” *Corpus technology and language pedagogy: New resources, new tools, new methods*, vol. 3, pp. 197–214, 2006.
- [7] D. S. Day, C. McHenry, R. Kozierok, and L. Riek, “Callisto: A configurable annotation workbench.,” in *LREC*, 2004.
- [8] J. Björne and T. Salakoski, “Tees 2.1: Automated annotation scheme learning in the bionlp 2013 shared task,” in *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 16–25, 2013.

- [9] P. Kolb, “Experiments on the difference between semantic similarity and relatedness,” in *Proceedings of the 17th Nordic Conference on Computational Linguistics-NODALIDA09*, 2009.
- [10] U. Consortium *et al.*, “The universal protein resource (uniprot) in 2010,” *Nucleic acids research*, vol. 38, no. suppl 1, pp. D142–D148, 2010.
- [11] D. Szklarczyk, A. Franceschini, M. Kuhn, M. Simonovic, A. Roth, P. Minguez, T. Doerks, M. Stark, J. Muller, P. Bork, *et al.*, “The string database in 2011: functional interaction networks of proteins, globally integrated and scored,” *Nucleic acids research*, vol. 39, no. suppl 1, pp. D561–D568, 2011.
- [12] A. Alawieh, Z. Sabra, M. Sabra, and F. Zaraket, “Novel bioinformatics approach reveals pathogenic mechanisms in cerebral ischemia-a step towards preclinical stroke information management system,” in *STROKE*, vol. 46, LIPPINCOTT WILLIAMS & WILKINS TWO COMMERCE SQ, 2001 MARKET ST, PHILADELPHIA, PA 19103 USA, 2015.
- [13] A. Alawieh, Z. Sabra, M. Sabra, F. Zaraket, and S. Tomlinson, “A rich club organization in spinal cord injury interactome provides insight into pathophysiological mechanisms and potential therapeutic interventions,” in *Spinal Cord Injury*, ASCI/AAP Joint Meeting, Chicago, IL, 2015.
- [14] A. Budanitsky and G. Hirst, “Evaluating wordnet-based measures of lexical semantic relatedness,” *Computational Linguistics*, vol. 1, no. 1, pp. 1–49, 2004.
- [15] C. Daly, W. P. Gibson, G. H. Taylor, G. L. Johnson, and P. Pasteris, “A knowledge-based approach to the statistical mapping of climate,” *Climate research*, vol. 22, no. 2, pp. 99–113, 2002.
- [16] C. Cardie, “Empirical methods in information extraction,” *AI magazine*, vol. 18, no. 4, p. 65, 1997.
- [17] P. Apache and A. Java, “To access microsoft format files,” 2009.
- [18] K. Boumillon and J. Levy, “Guava: Google core libraries for java 1.5+.”
- [19] H. A. Whiteford, L. Degenhardt, J. Rehm, A. J. Baxter, A. J. Ferrari, H. E. Erskine, F. J. Charlson, R. E. Norman, A. D. Flaxman, N. Johns, *et al.*, “Global burden of disease attributable to mental and substance use disorders: findings from the global burden of disease study 2010,” *The Lancet*, vol. 382, no. 9904, pp. 1575–1586, 2013.

- [20] A. S. Go, D. Mozaffarian, V. L. Roger, E. J. Benjamin, J. D. Berry, W. B. Borden, D. M. Bravata, S. Dai, E. S. Ford, C. S. Fox, *et al.*, “Heart disease and stroke statistics–2013 update: a report from the american heart association.,” *Circulation*, vol. 127, no. 1, p. e6, 2013.
- [21] B. V. Zlokovic and J. H. Griffin, “Cytoprotective protein c pathways and implications for stroke and neurological disorders,” *Trends in neurosciences*, vol. 34, no. 4, pp. 198–209, 2011.
- [22] D. W. Huang, B. T. Sherman, and R. A. Lempicki, “Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists,” *Nucleic acids research*, vol. 37, no. 1, pp. 1–13, 2009.
- [23] A. Franceschini, D. Szklarczyk, S. Frankild, M. Kuhn, M. Simonovic, A. Roth, J. Lin, P. Minguez, P. Bork, C. von Mering, *et al.*, “String v9.1: protein-protein interaction networks, with increased coverage and integration,” *Nucleic acids research*, vol. 41, no. D1, pp. D808–D815, 2013.
- [24] B. Demchak, T. Hull, M. Reich, T. Liefeld, M. Smoot, T. Ideker, and J. P. Mesirov, “Cytoscape: the network visualization tool for genomespace,” 2014.
- [25] M. Kuhn, D. Szklarczyk, A. Franceschini, C. von Mering, L. J. Jensen, and P. Bork, “Stitch 3: zooming in on protein–chemical interactions,” *Nucleic acids research*, vol. 40, no. D1, pp. D876–D880, 2012.
- [26] K. Konganti, G. Wang, E. Yang, and J. J. Cai, “Sbtoolbox: a matlab toolbox for biological network analysis,” *Evolutionary bioinformatics online*, vol. 9, p. 355, 2013.
- [27] V. Colizza, A. Flammini, M. A. Serrano, and A. Vespignani, “Detecting rich-club ordering in complex networks,” *Nature physics*, vol. 2, no. 2, pp. 110–115, 2006.
- [28] F. Viger and M. Latapy, “Efficient and simple generation of random simple connected graphs with prescribed degree sequence,” in *Computing and Combinatorics*, pp. 440–449, Springer, 2005.
- [29] D. W. Huang, B. T. Sherman, and R. A. Lempicki, “Systematic and integrative analysis of large gene lists using david bioinformatics resources,” *Nature protocols*, vol. 4, no. 1, pp. 44–57, 2008.
- [30] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, “Cytoscape: a software environment for integrated models of biomolecular interaction networks,” *Genome research*, vol. 13, no. 11, pp. 2498–2504, 2003.

- [31] D. Tabas-Madrid, R. Nogales-Cadenas, and A. Pascual-Montano, “Genecodis3: a non-redundant and modular enrichment analysis tool for functional genomics,” *Nucleic acids research*, vol. 40, no. W1, pp. W478–W483, 2012.
- [32] C. Gibson, S. Turner, and M. Donnelly, “One degree of separation: paralysis and spinal cord injury in the united states,” *Christopher and Dana Reeve Foundation, Short Hills*, 2009.
- [33] B. T. Sherman, D. W. Huang, Q. Tan, Y. Guo, S. Bour, D. Liu, R. Stephens, M. W. Baseler, H. C. Lane, and R. A. Lempicki, “David knowledgebase: a gene-centered database integrating heterogeneous gene annotation resources to facilitate high-throughput gene functional analysis,” *Bmc Bioinformatics*, vol. 8, no. 1, p. 426, 2007.
- [34] C. Von Mering, M. Huynen, D. Jaeggi, S. Schmidt, P. Bork, and B. Snel, “String: a database of predicted functional associations between proteins,” *Nucleic acids research*, vol. 31, no. 1, pp. 258–261, 2003.
- [35] P. Stenetorp, S. Pyysalo, S. Ananiadou, and J. Tsujii, “Almost total recall: Semantic category disambiguation using large lexical resources and approximate string matching,” in *Proceedings of the Fourth International Symposium on Languages in Biology and Medicine*, 2011.



# Appendix A

## Tutorial

### A.1 How to run the tool

SAMNA is a java tool. It is uploaded on Google svn under the name "medannotation" <https://code.google.com/p/medannotation/source/browse/#svn%2Ftrunk> Specialist can run SAMNA by:

- building the source code and get the jar file, then
- run the jar file by the command "java - jar JarFileName.jar or java -Xmx8096m -jar JarFileName.jar". -Xmx8096m is used to give the tool more RAM memory in case of large data file.

Both the jar file and the command can be given to the medical specialist in one folder. The specialist has to double click on the command icon which will run the jar file. NOTE: SAMNA can be imported as a Netbeans project.

### A.2 Tool Prerequisite

When the specialist run SAMNA it asks him to specify the work space directory.(Figure A.1)

To properly run SAMNA specialist should choose the work space directory where he has to have:

1. The protein database named : "formoh-labels.xlsx"
2. The result database named: "results.xlsx"
3. The local non protein database named "LOCALNonProtein.xlsx"

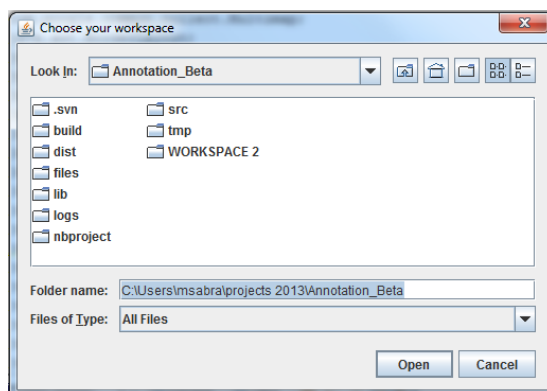


Figure A.1: Choosing work space when you first run the tool

4. The labels database named "Labels.xlsx"
5. The excluded abstract database named "exludedAbstracts.xlsx"

## A.3 Tool Features

After choosing the work space SAMNA open in two stage.

1. Loading frame where specialist has to choose the type of loaded file.
2. Main frame where the specialist can annotate the loaded file.

### A.3.1 Loading frame

After choosing the work space SAMNA will open the loading frame (Figure A.2) where the specialist has to choose between:

1. Loading abstract fro a serialized object.
2. Loading abstracts from XML file.
3. Loading abstract from a CSV file
4. loading auto extracted paper
5. Loading abstract from a text file.

After choosing the file to load, the Main frame of SAMNA will open (Figure A.3)

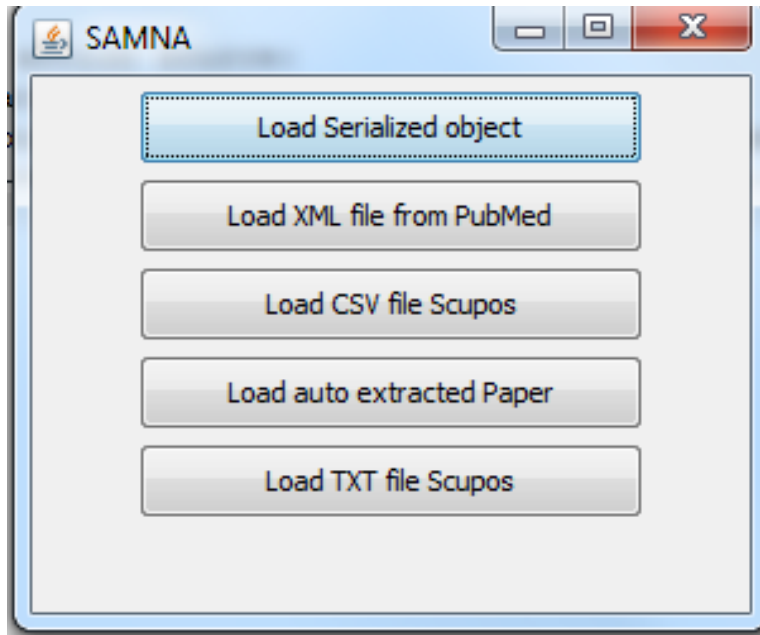


Figure A.2: SAMNA Loading frame

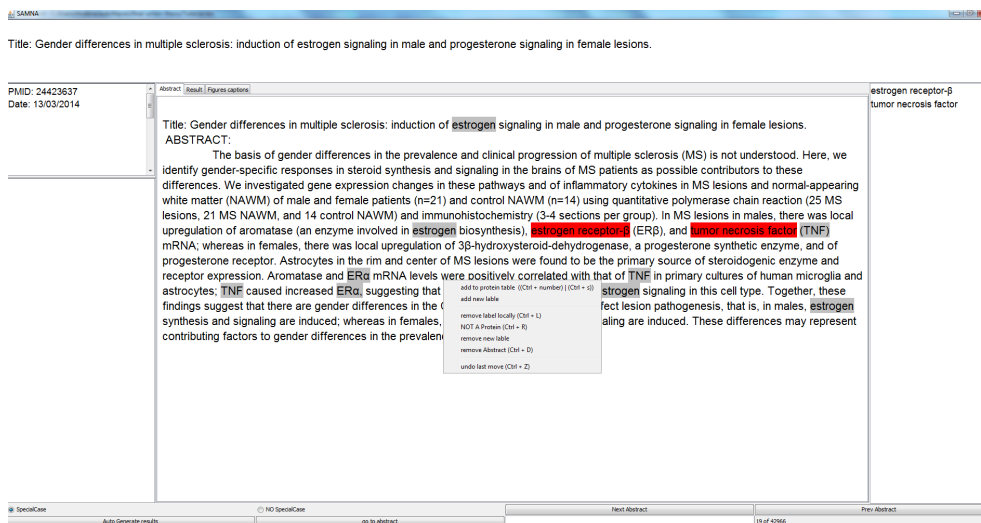


Figure A.3: SAMNA main frame

### A.3.2 Main frame

In the main frame SAMNA has two action source:

1. The buttons line on the bottom of the frame, which contain buttons from left to right that allow the specialist to:
  - Choose if he wants to apply the knowledge base rules.

- Go between abstract forward and backward.
  - Go through all abstract automatically (auto generate result button).
  - Go to a special abstract number directly.
2. The right click mouse pop up and its shortcuts which allow specialist to:
- Add new annotation to the label database.
  - Add new label other than the protein default label.
  - Add new annotation to the new label database, when specialist adds a new label, a new option will appear that allow the user to add an annotation to the new label.
  - Remove an added label.
  - Remove a annotation locally, if the specialist want to not annotate a term if this term is founded in a specific abstract.
  - Remove a protein annotation from the database.
  - Remove an annotation from its newly created label database.
  - Remove an abstract, if the specialist want to not annotate terms in a specific abstract.
  - Undo important actions.

## A.4 Loading Abstracts

The first step Specialist has to do with SAMNA is to load the abstracts into the frame. The abstracts could be in an XML file extracted from PubMed database through the The National Center for Biotechnology Information (NCBI) website, in a TXT and CSV file extracted from Scopus database, in a folder format containing the auto extracted papers retrieved by a special script along with SAMNA, and in a serialized object format which is an object containing the needed data to load.

When loading the abstracts into the frame (Figure A.4) specialist can notice

1. Number of abstracts and the abstract currently loaded.
2. Title section where the title of the abstract is shown.
3. PMID and Date section where the PMID and date of the loaded abstract is located.
4. The highlighted word section where the annotations are listed.

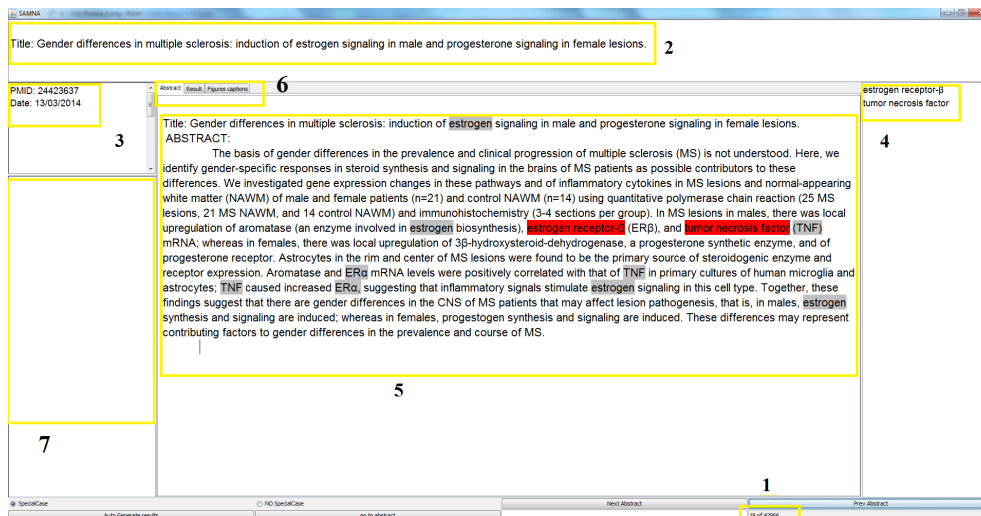


Figure A.4: loading Abstract Picture

5. The main frame section where the title and the abstract text are annotated and the suggestions are highlighted in gray.
6. The tabs section, when the specialist loads auto extracted paper he will have a result section and a figures captions section along with each figure in a tab.
7. Distribution section where the specialist can see each annotation distributions in the file if he clicked on the annotation in the highlighted word section.

### A.4.1 Loading Abstracts from an XML file

SAMNA is designed to load XML file that contain medical publications, <PubmedArticle>node for each publication inside this "PubmedArticle" node we have 4 important nodes to be extracted and shown in the GUI:

1. <PMID>node that contain the publication Id.
2. <DateCompleted>node that contain the Date of the articles.
3. <AbstractText>which is the most important node that contain the abstract of the articles.
4. <ArticleTitle>that contain the title of the article.

To load an XML file specialist has to click on "load XML file from PubMed" button which will open for him a file chooser dialog to choose the XML file he wants to load.

After choosing the XML file, SAMNA will parse this file, put all article in a list, specialist can see each abstract alone with its title, PMID, and date.

When Loading the XML file into the frame SAMNA creates a serialized object in the same directory of the file, this serialized object contain the list of abstracts already loaded from the XML file.

### **A.4.2 Loading Serialized Abstracts**

The serialized object is created when loading the abstract to minimize the loading time. Because the XML file could have a large size (800 MB), that could cost hours to be loaded.

After loading the file the serialized object will contain only elements of interest to the specialist (Abstract, Title, PMID, Date). The serialized object will have a much lower size (10% of the original file size). The serialized will cost much less time to be loaded ( 30 minutes to load a file with 330 MB size, will be 5 seconds to load its serialized object with 30 MB size).

If the XML was large specialist will only pay the time to open it the first time, the second time he could load it from the serialized object and it will coast him much less time.

To load the abstracts from the serialized object specialist has to click the button "Load serialized object" which will open for him a file chooser where he can choose the serialized object.

Note that the serialized object will have the same name as the xml file but with different extension(".ser").

### **A.4.3 Loading Abstracts from TXT and CSV file**

As above specialist can also load abstract from a TXT and CSV file extracted from Scopus, which has some special format. To load abstract from TXT and CSV file specialist has to click on the "Load TXT file scopus" button or "Load CSV file scopus" button. and then choose the text or CSV file.

## **A.5 Moving between Abstracts**

After loading the abstracts, specialist can move between abstracts by clicking on "Next Abstract", "Prev Abstract", "Auto Generate results", and "go to abstract" buttons.

While going between abstract each term founded in the predefined database will be annotated and added to the result database along with its PMID and Date. Specialist can automatically go through all abstract by clicking "auto generate abstract" button, and he can go to a specific abstract by putting its number and

click "go to abstract" button.

Next and Prev abstract button are made for the specialist to go between abstracts, so he can add a annotation to its label database, remove an annotation if he realize that this term should not be in the label database, or should not be considered as a label in this special abstract. Specialist can also add some label other then the default label so he can construct a new label database from the abstracts.

## A.6 Adding Annotation

the tool allow specialist to add a protein to the protein database (Figure A.5)

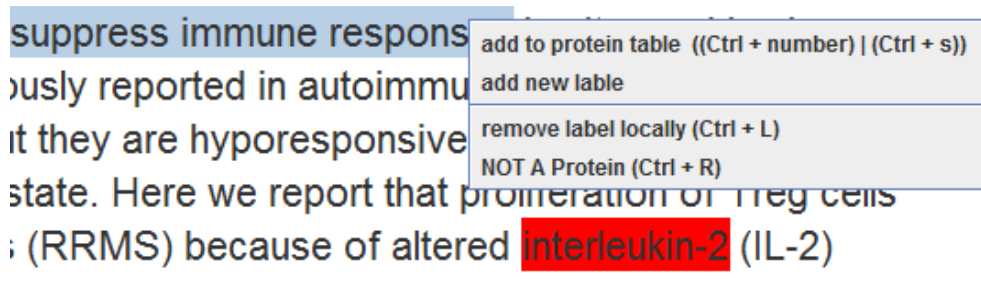


Figure A.5: Adding protein to database

all he has to do is to:

1. Select a word he wants to add.
2. Right click add click add to protein table.

after he does those 2 steps the protein will be added to its database, and the term will be annotated in all abstracts in a special color (yellow). Note: to add a protein specialist can use two shortcut Ctrl+s to add the selected words, or Ctrl+ number to add a number of word from the selected words to the database. (ex: CTL+5 will add the last 5 words of the selected words to the database).

## A.7 Removing Annotation

The figure 4 shows that the specialist can also remove selected protein from its database (Not A protein) or not consider it as protein in a particular abstract (remove label locally) those two options have shortcut: Ctrl+R and Ctrl+L respectively.

## A.8 Apply expert Rules

While moving between abstracts specialist can choose to consider special case or not to consider them, Special case are the cases where SAMNA will apply specialist rules.( ex: protein named "XYZF" and we have a word "XYZF-5" if specialist did choose to consider special case SAMNA will highlight the word "XYZF-5" for him, otherwise it will not highlight it).

## A.9 Distributional Similarity suggestion

While moving between abstracts the tool will suggests new protein name for the specialist in a gray color so he can pay attention of the probability to have a new annotation that should be added.(Figure A.6)

gene expression profile of thoracic propriospinal neurons between  
ided from GEO database, including 12 Spinal Cord Injury (SCI) rat  
y expressed genes with a fold-change > 1.2. Then, we used DAVID  
l biological processes and molecular signatures database (MsigDE  
gnature and other published gene expression signature. Protein-Pro  
Cytoscape. Functional analysis of the hub protein was performed  
ofile was found at 3-days post injury and immune response was fc  
weeks post injury was found significantly overlapped with genes upr  
network analysis found that LYN, PTPN6 and SMAD1 could be of  
derlying molecular mechanism post injury, especially at early mom

Figure A.6: suggestion base on the distributional similarity algorithm

## A.10 Adding New Labels

SAMNA used first to extract protein name from abstract, add protein to database can be used for any element other than protein, specialist is able to add a new label, name it, then choose a color to highlight with. by clicking on "add new label after right click" then the specialist will be able to choose a name and color for this new label (Figure A.7).

After adding a new label a new right click action will be added, to allow adding annotation to the new created label.



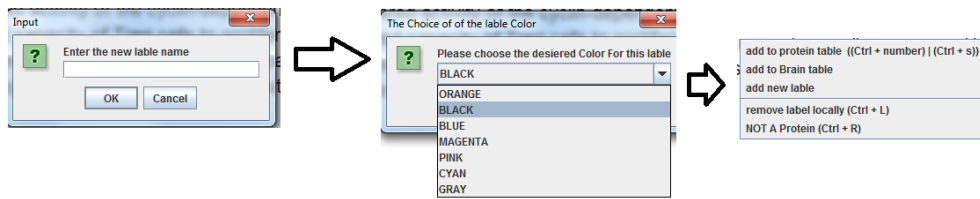


Figure A.7: Adding new label

## A.11 Annotation distribution

Specialist can see each annotation distribution when he double click on the annotation in the annotations list.(Figure A.8). SAMNA will take some time to go through all abstracts, but after this the specialist can have an idea about the distribution of each annotation in the file.

Figure A.8: annotation distribution

## A.12 Exclude Abstract

SAMNA allows the specialist to exclude an abstract so the annotation in this abstract will not be considered in the result file. Excluding abstract can be done by right click and press "remove abstract" or shortcut it by Ctrl+D Excluded abstract will be disabled as shown in (Figure A.9).

## A.13 Undo Actions

SAMNA allow the specialist to undo any action he made while he is still in the abstract, he just has to right click and press undo or press the shortcut CTRL+Z, if he did add some annotation it will be deleted from the database, if he deletes some annotation it will be added again (Figure A.10).

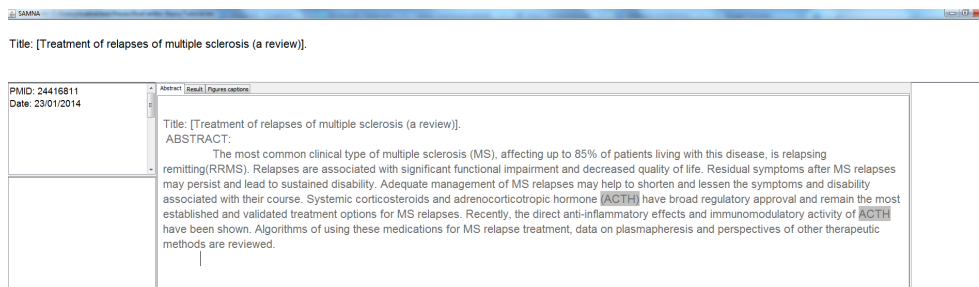


Figure A.9: removing abstract

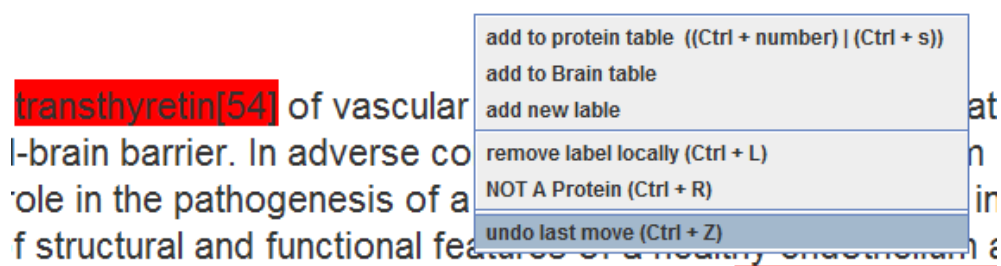


Figure A.10: undo an action