CIS 700B Master's Project Report

Developing Health Knowledge Refinery Portal System

Submitted to
The Department of Computer Science,
New Jersey Institute of Technology.

In Partial fulfillment of the requirements of the Degree of Master of Science in Computer Science

Submitted by **SHRIKANT DOIPHODE**
NJIT ID: 217-71-344
sd269@njit.edu

Project Advisor: Dr. James Geller

APPROVALS

Proposal Number: _____

Agree to Advise: __**Prof. James Geller**_____
(Project Advisor)

Date Submitted: _____

Approved by: _____
(MS in CS Committee)

Date Approved: _____

Signature: _____
(Shrikant Doiphode)

Abstract

The health care horizon is rapidly changing with the Internet and the vast amount of health information contributed not only by professional health care providers (doctors, nurses, etc.) or authoritative research results, but also by many patients. Patients, who used to rely on health care experts for advice and treatments, now try to seek health information on their own from contents by other patients via social media, e.g. reading patients blogs, discussion groups, and social networking sites. However, the Internet's content diversity is often overwhelming, because users are not aware of the full extent of information available on the Internet. With the ever increasing text information offered by social media-based health and medical discussion groups, it becomes even more difficult to find the right place for seeking to fulfill a specific medical information need. Patients or care givers need to have access to an integrated knowledge portal that provides them with aggregated health information to understand the diseases, alternative treatments and side-effects of drugs as well as to analyze and understand trends and anomalies for better health care.

This project addresses how to link scattered health-related data from different Web communities, and provide integrated knowledge of health information (symptoms, treatments, side-effects). Specifically, we integrate data from social media-based patients' communities, the clinical experts' community and the research community. We utilized content extraction, content organization and summarization of health discourse in social media Websites, including blogs and health support discussion groups, as well as knowledge provided in the professional and research community. The content extraction component utilizes text extraction and text mining technology to extract the key health terms from the unstructured text found in various health care social media. The summarization component aggregates patient-contributed content to show the

social population trends in health care. These enable users - (patients, doctors or any health care providers) to search health-related terms and view related information in a summarized manner. This may include the symptoms specified by the patients, drugs and treatments specified by recognized doctors or health care organizations, effects of specific drugs or treatments discussed by patients and comments posted by either patients or authenticated doctors. A prototype system called "Health Knowledge Refinery Portal" has been developed to identify the health-related terms occurring in a blog and annotate them with the linked and aggregated health knowledge extracted from social networks, specifically from Patients-like-me, and from the WebMD and the PubMed sites.

Table of Contents

1. Introduction

Nowadays use of the Internet for health care purposes by health care professionals (clinicians) and also, more interestingly, by lay persons (patients) is ever increasing. A recent survey in seven European countries has shown that on average 69% of the Internet users occasionally use the Internet for health care purposes. They use the Internet for various purposes such as self-help activities, reading about health or illnesses, and querying about diagnoses. Health care professionals or service providers deliver health-related information in the form of consumer-oriented Web resources, and lay people search and access it through the Internet to get some useful insights.

1.1 Need of Health Knowledge Refinery Portal

We can get good information for health care purposes on the Internet, but most of the times this information is not so concise, or only a part of the whole document is related to our interests. Health knowledge is widely distributed over the Internet. Users have to put in the effort to gain detailed knowledge of a particular drug or symptom. They have to visit many different Websites, for example www.patientslikeme.com, www.Webmd.com, www.pubmed.com or read the whole document or page to gather the information about a particular drug or symptom. This information is neither presented in a summarized manner nor does the information on a blog provide any statistics.

Thus, there is a definite need of a system that integrates most of the information about the drugs and symptoms from the different sources as those mentioned above, calculates the statistics and displays all this in the interface of a single independent system. This will help the users for

understanding the trends in medicine and make more informed decisions with integrated knowledge for their health care purposes.

Below are a few examples of health data that is distributed, incomplete and inadequate. The users have to make an extra effort to get a better understanding from this health data.

1.1.1 Use of Social Media for Medical Discussions

1. Data from Patients Like Me Website:

   http://www.patientslikeme.com/all/symptoms/show/575-headaches

This site specifies the symptoms of headaches faced by patients. It shows a pie-chart focusing on the severity of symptoms and also the number of patients taking a particular medicine for the treatment of headache.

> *Example 1:*
> *Ibuprofen 129, Acetaminophen 68, Topiramate 66, Excedrin Migraine 29, Butalbital-acetaminophen-caffeine 26, Chiropractic 24, Midrin 15, Amitriptyline 14, Hydrocodone-Acetaminophen 14, Naproxen Sodium OTC 14, Propranolol 13, Tramadol 13, Aspirin 12.*

The www.patientslikeme.com, Website provides a list of all the drugs that are used for treating headache. It doesn't provide any description about that drugs, rather it provides only a list of the drugs.

This data does not help us to draw a proper conclusion such as which medicine is the best for treating headache, because it does not focus on ***statistics*** (i.e. percentage of patients taking a particular drug), ***side effects*** of the drug, ***severities of the side effects***, ***reasons for stopping*** the drug consumption, etc.

2. Data from WebMD Health Blog:

   http://exchanges.Webmd.com/breast-cancer-exchange

This is a WebMD breast cancer community. The questions asked here in the discussion board are answered by a doctor, Dr. Philomena McAndrew.

Here is an example from the discussion board on breast cancer:

> *Example 2:*
> *I am 6 months cancer free from DCIS. They want me to take Tamoxifen but I smoke. I am afraid to take it because of that. While I am trying to quit smoking, is taking this drug going to harm me even more because I smoke?*
> *Dr. Philomena McAndrew.*
> *Yesterday on "The View" they had a discussion on mammograms. Two of the ladies had their mammos done and invited the audience along. They showed some of the prep for it, and I noted one big difference from those I have had over the years: They each were given a shield for thier lower bodies. Seems a good idea that is late in coming. Is this now the practice in most image centers?*
> *Oh, yes, one other thing: They had a doctor on who wasn't the best informed. When asked if there was any way to ease the discomfort during the mammogram, he seemed to offer no real suggestions (ie: time in your menstral cycle, avoiding caffeine for a week or so prior, asking for the new paddings available, etc.). Instead he stressed it was worth the little discomfort. I've no argument with his weighing the value of the test, but sure wish he had offered more concrete information!*

We have to read the whole document to extract the useful health-related information. This information is based on user experience and may also be incomplete.

3. Data from PubMed:

http://www.ncbi.nlm.nih.gov/pubmed?term=headache%20treatment

Pubmed.gov is a Website of the U.S. National Library of Medicine National Institutes of Health. It is a collection of health-related research papers. For example, here are papers about headache.

1. *Impaired glutamatergic neurotransmission in migraine with aura? Evidence by an input-output curves transcranial magnetic stimulation study.*
   Cosentino G, Fierro B, Vigneri S, Talamanca S, Palermo A, Puma A, Brighina F.
   Headache. 2011 May;51(5):726-33. doi: 10.1111/j.1526-4610.2011.01893.x.

2. *Sex Differences in Behavior and Expression of CGRP-Related Genes in a Rodent Model of Chronic Migraine.*
   Stucky NL, Gregory E, Winter MK, He YY, Hamilton ES, McCarson KE, Berman NE.
   Headache. 2011 May;51(5):674-92. doi: 10.1111/j.1526-4610.2011.01882.x.

3. *Sumatriptan-Naproxen Sodium for Menstrual Migraine and Dysmenorrhea: Satisfaction, Productivity, and Functional Disability Outcomes.*
   Cady RK, Diamond ML, Diamond MP, Ballard JE, Lener ME, Dorner DP, Derosier FJ, McDonald SA, White J, Runken MC.
   Headache. 2011 May;51(5):664-673. doi: 10.1111/j.1526-4610.2011.01894.x.

4. *Management of idiopathic intracranial hypertension in parturients: anesthetic considerations.*
   Karmaniolou I, Petropoulos G, Theodoraki K.
   Can J Anaesth. 2011 Apr 26. [Epub ahead of print]

It provides a list of research papers published and if related to headache (user specified), the user needs to read each of these research papers to get information about headache.

Many of the research papers have limited accessibility for reading (as they might require payment), so users can only access abstracts of such papers. It is very difficult to extract useful information from all these research papers, as they are very hard to understand and often assume excellent medical background of the reader.

1.2 Challenges for Developing the Health Knowledge Refinery Portal

1) Social media-based health information discourse is low in usefulness without content summarization. Just looking at the second example in Section 1.1.1, the data is so sparse that a study on text extraction to organize the content is needed.

2) The conversation threads involve several participants and often content builds on previous discussions by others. A user who wants to summarize all the information has to look at each thread and its responses. But if we develop a discourse representation with tracking "who said what with which supporting arguments or examples," that would ease the use a lot.

3) The users have a hard time or spend a lot of time to get a concise summary of the available information. For this reason, a search, summary and recommendation support system is needed.

4) Since the information used for this project is written mostly by lay persons, there is no standard format in expressing what the information is about or in what terminology or concepts are used. The information could be irrelevant with respect to the health/medical domain or invalid or mismatched with the standard medical terminologies. Nevertheless, we observe that many users (i.e. patients) believe that they can extract useful information from these Web resources. As an example, a survey has suggested that blog users study cancer blogs, for seeking compiled cancer information, expanding their cancer knowledge, seeking information to help friends and family and even validating information received from their health care providers.

Clearly, unreliable or incomplete information could have serious consequences for its users – the effective usefulness of social media-based discussions and blogs is low due to the poor organization of contents. The online conversations are often unstructured and the readers need to spend a lot of time reading the conversation threads to gather useful information. The content is hard to follow without summarizing the discussions among participating patients and health professionals. There is as yet little insight into the extent to which sites offer their users information that is easily found, well understood and relevant to their needs.

As shown in the examples above, it is difficult to grasp the meaningful information in a dialog or a description. The users need to read all the conversation turns and threads to gain some useful pieces of information. It would be more useful if the content of discourse (conversations) is presented in a summarized manner. In other words, useful information needs to be extracted from the health-related social media and presented in a summarized manner.

To address the above problems, we propose a system called ***Health Knowledge Refinery Portal,*** which can be used to extract health-related keywords from blogs or any health Web portal or health-related Website, and provide the brief summary for each matched keyword along with the

statistical information related to the keywords, linking scattered health information and providing integrated knowledge derived from social, clinical and research information on a specific health topic.

1.3 Overview of Health Knowledge Refinery Portal

After having established the need for a portal in section 1.1, we can conclude that the textual information in medical blogs is very verbose and unstructured, and important information or points are fragmented in different parts of discourse. It would be time consuming to read all the conversations in order to gather the needed information. Thus, I have developed an integrated system, ***Health Knowledge Refinery Portal*** which extracts all the symptoms or treatments from the Health Blog specified by the user. This system will highlight all the symptoms or treatments found in the text extracted from the Health Blog URL specified by the user. These highlighted keywords, when clicked, will display nice summarized information about the same. A summary consists of a brief description of the keyword and category, along with the statistics in tabular format. I also provide WebMD and PubMed external links as source for expert knowledge and research.

1.5 Outline

In section 2, we discuss background literature about a few primary approaches towards the problem addressed in this project. Section 3 presents the actual approach towards the implementation of the project, along with the ***Health Knowledge Refinery Portal*** system architecture. Section 4 presents the prototype system of the project, i.e., the actual functionality of the ***Health Knowledge Refinery Portal***. Section 5 presents the conclusions after implementing the ***Health Knowledge Refinery Portal.***

2. Literature Survey

2.1 Approaches to Text Extraction

2.1.1 First Approach: Using KNN

Akbar, Slaughter and Nytro used K-Nearest Neighbor classification for collecting health-related emails from a breast cancer mailing list. They experimented with different features and observed which features are preferable. The experiments revealed that UMLS terms extracted from emails could constitute a good feature. In contrast, health-related UMLS semantic types as a feature did not show a good accuracy [3].

2.1.2 Second Approach: Using CaRE and GATE

Doctors providing care for their patients need to know what symptoms the patient has, allergies, drugs taken, tests ordered, etc. Researchers trying to find statistics on patients with particular diseases or symptoms need a more efficient way to find all these records. The problem is that this information is not readily available. If the information is digitized and formatted appropriately it can then easily be queried and prevent medical errors caused by miscommunication.

Many programs have been developed to help with this problem. Tawanda Sibanda developed the Category and Relationship Extractor (CaRE), which finds semantic categories and their relationships in medical discharge papers [6]. The tools developed by Sibanda can be integrated so that they work together seamlessly. These tools include preprocessing programs for Assertion Classification, De-identification, Concept Recognition, and Relationship Extraction, along with Support Vector Machine (SVM) training and prediction programs. These tools were incorporated into the General Architecture for Text Engineering (GATE), a free software framework available online. However, the GATE application's run time was poor for large files.

2.2 Industry Tools to Extract Data

Mozenda is a Software as a Service (SaaS) company that enables users to easily and affordably extract and manage Web data. With Mozenda, users can set up agents that routinely extract data, store data, and publish data to multiple destinations. Once information is in the Mozenda system, users can format, repurpose, and mash up the data to be used in other online/offline applications or as intelligence. All data in the Mozenda system is secure and is hosted in a class A data warehouses but can be accessed over the Web securely via the Mozenda Web Console. With the addition of the fully featured REST API of Mozenda, companies can now seamlessly integrate their data automation with the Mozenda application. Unfortunately, Mozenda is expensive ($99/month).

Extract Transform Load (ETL) is a common terminology used in data warehousing, which stands for extracting data from source systems, transforming the data according to the business rules of the organization and loading it into the target data warehouse. In the early days of data warehousing, this process was done by writing complex code, which was an inefficient way to process large volumes of complex data in a timely manner. This approach is good for structured data but not suitable for unstructured text data.

3. Our Approach

The ***Health Knowledge Refinery Portal*** we have developed consists of the following components:

1) Social Health Knowledge Base: This component uses Web data extraction and data mining to extract the rich social health knowledge on treatments, symptoms, and side effects, contributed by patients, and construct a health knowledge base from it. This social health knowledge is valuable since it is based on a collection of actual patients' experiences. This empirical data from a large number of patients in a social network can be used as an indicator of the efficacy of the prescribed treatments or drugs. Collecting this kind of patients' usage data has not been easy, thus it is not readily available for an automated system to utilize. It is often collected by governments through reports by doctors or health providers based on interactions with patients or by large pharmaceutical companies through clinical experiments

2) Blog Extraction Component: To illustrate how to support the users with the social health knowledge base extracted from social health networks, we developed a component to identify health terms in a user input health blog. This blog extraction component extracts all the text from the blog URL specified by the user using the ***jsoup*** API[1]

3) Health Keywords Identification component: Once the text is extracted from a designated blog, the Health Keyword Identification component looks for the health-related keywords that matched with the keywords (drugs or symptoms) stored in the Social Health Knowledge Base.

[1] http://boilerpipe-web.appspot.com/

4) Integrated Health Summary and Linking component: All the matched heath-related keywords i.e. either ***Drugs*** or ***Symptoms*** are marked up with health summary data from the Social Health Knowledge Base and with hyperlinks to WebMD and PubMed medical knowledge relevant to the health keywords, providing an integrated health and medical information environment.

Any user that wants to access the system has to provide a valid URL. Users can select a URL from a predefined list of health discussion URLs or they can enter a URL that they want to analyze. Users may select "drugs" or "treatments" as search criteria and as a result get all the available relevant information in a summarized manner. This information may contain the past experience of patients and effects of drugs/treatments observed by them.

Users can click on any keyword to get summarized information of that keyword from our knowledge base. The summary for any matched keyword consists of a ***detailed description*** of the keyword and ***statistics*** featuring ***percentages*** of patients taking a particular drug, ***side effects*** of the drug, ***severities of the side effects***, ***reasons for stopping*** the drug consumption, etc.

Percentages of patients taking a particular drug for a common symptom constitute one of the important factors to decide what would be the best drug among the list for treating a specific symptom. Side effects of consuming the particular drug along with the statistics and severities of the side effects help us to get adequate knowledge about that particular drug. Reasons for stopping the use of a particular drug helps users know whether they should consider stopping the drug consumption.

Figure 1 shows overall system architecture with the external users.

System Architecture:

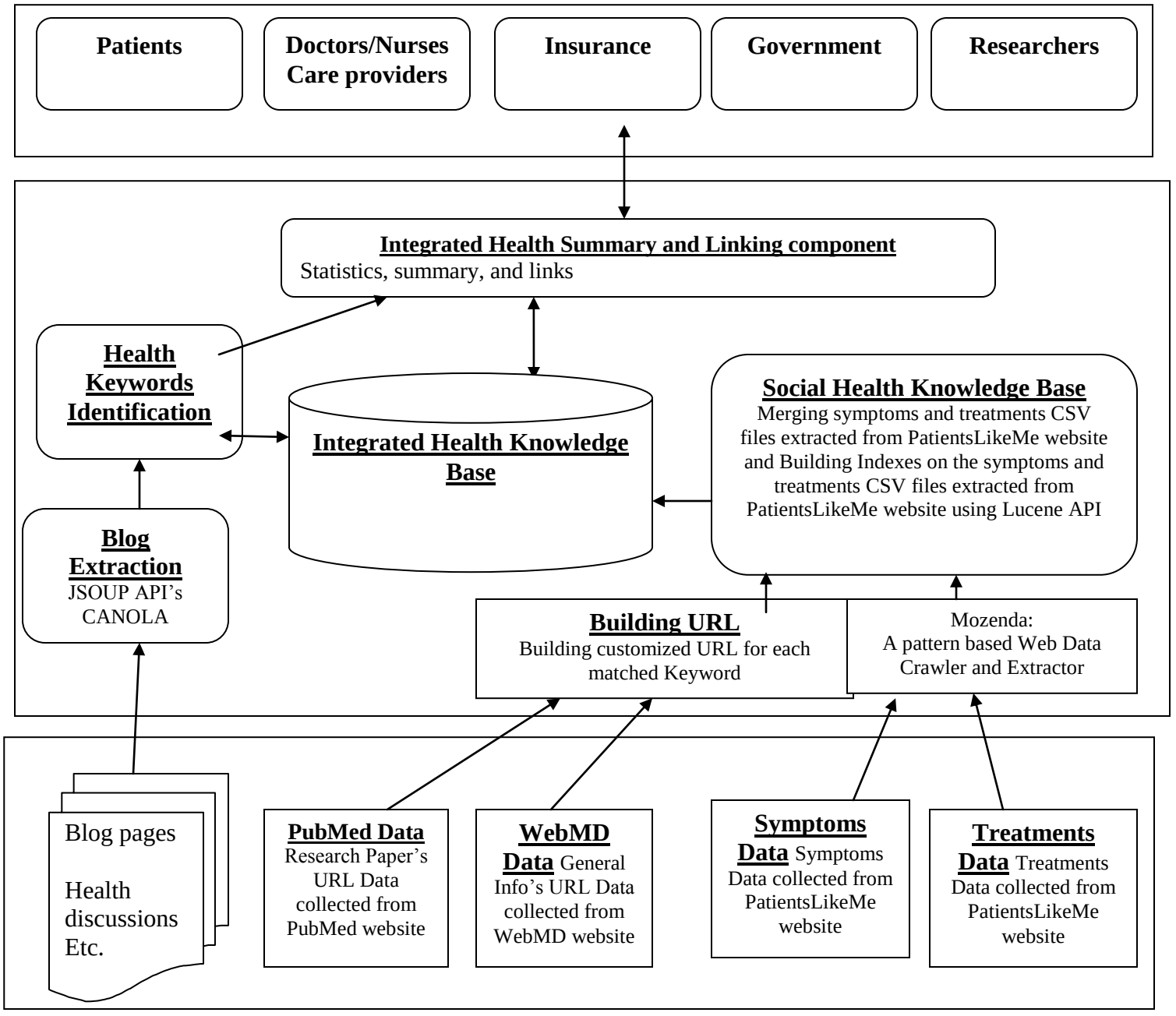

Figure 1: System Architecture for Health Knowledge Refinery Portal

4. Prototype System

We have developed a prototype system of the Health Knowledge Refinery Portal. The overall system data and process flow are shown in figure 2.

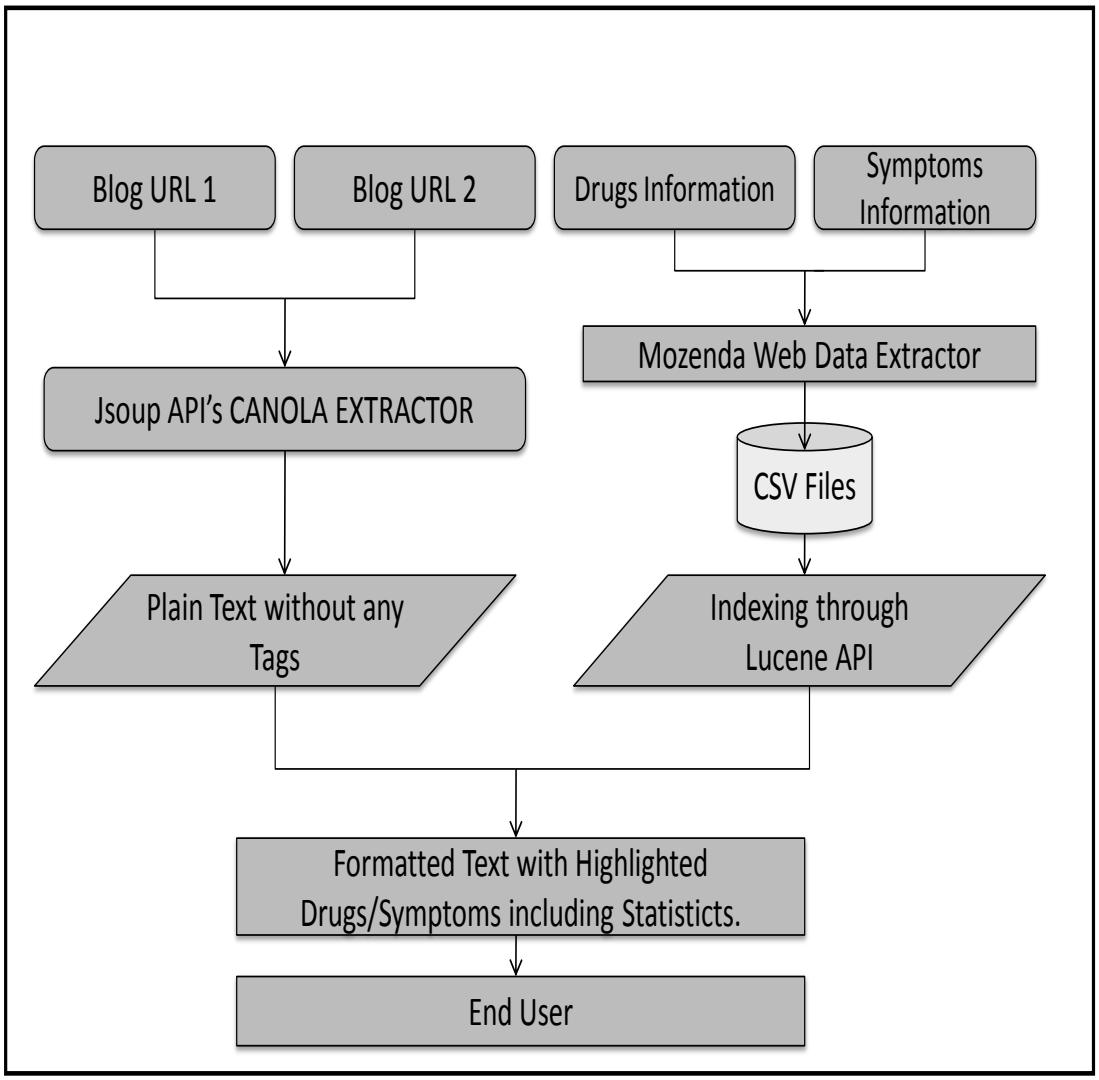

Figure 2: Flow Diagram for Health Knowledge Refinery Portal

4.1 Module 1: Social Health Knowledge Base

The Social Health Knowledge Base module collects data from the www.patientslikeme.com Website using an external tool, Mozenda[2]. The data collected from www.patientslikeme.com consists of drug data and symptom data. This data is then passed to the Lucene API, to perform indexing. Indexes are created on drug names and symptom names. These indexes are then used to match the contents in the plain text extracted in Module 1.

To construct the knowledge base, we have used data from:

- **PatientLikeMe** Social Network Site**:** PatientsLikeMe Website has mainly categorized its data by symptoms and treatments.
- **Pubmed:** Pubmed is a US National Library of Medicine database. It is a library that stores most English medical research papers. There are many research papers on symptoms and drugs. So, I constructed an URL for accessing the research papers in Pubmed for all the symptoms and treatments that are matched in the text extracted from the URL specified by the user.
- **WebMD** – WebMD is a repository that describes the top news and information for each specified medical keyword. Thus, I constructed an URL for accessing all the information from WebMD for the symptoms or treatments that are matched to the text extracted from the URL specified by the user.

[2] Mozenda is Software as a Service (SaaS) company that enables users to easily and affordably extract and manage Web data

As described, I collected the data from various sources like www.patientslikeme.com, www.Webmd.com and www.pubmed.com. The data collected from www.patientslikeme.com mainly consists of symptoms and drug data with all the relevant information associated with them. I used the Mozenda text extraction tool to extract the data from www.patientslikeme.com because Mozenda is a Software as a Service (SaaS) company that enables users to easily and affordably extract and manage Web data. With Mozenda, users can set up agents that routinely extract data, store data, and publish data to multiple destinations. Once information is in the Mozenda system, users can format, repurpose, and mash up the data to be used in other online/offline applications or as intelligence. The data collected from www.Webmd.com and www.pubmed.com consists of external link data, which is displayed to the user whenever the user clicks on any matched keyword.

Mozenda is a tool that extracts pattern-based data from all the pages in a list; it will crawl through the Website according to the pattern that users specify. While crawling, it will extract all the data as per the pattern and eventually export all the extracted data as a CSV file.

For extracting the data from PatientsLikeMe Website I specified two patterns:

1) For symptoms data, I collected the data in the following pattern:

Symptom Name, Number of Patients Suffering, Brief Description of Symptom, Which Drug did Patient Use to Treat the Symptom, Number of Patients Taking the Drug

Figure 3, shows the algorithm for extracting the symptoms data from the PatientsLikeMe Website. Mozenda extracts the text as per the above pattern from the whole list of pages as shown in figure 3.

It will extract *Symptom Name, Number of Patients Suffering, Brief Description of Symptom, Which Drug did Patient Use to Treat the Symptom, Number of Patients Taking the Drug,* in a single iteration then it will check for the end of the list. If the list ends, then Mozenda will export the extracted text as a CSV file. If the list does not end, it will continue its extraction iteration.

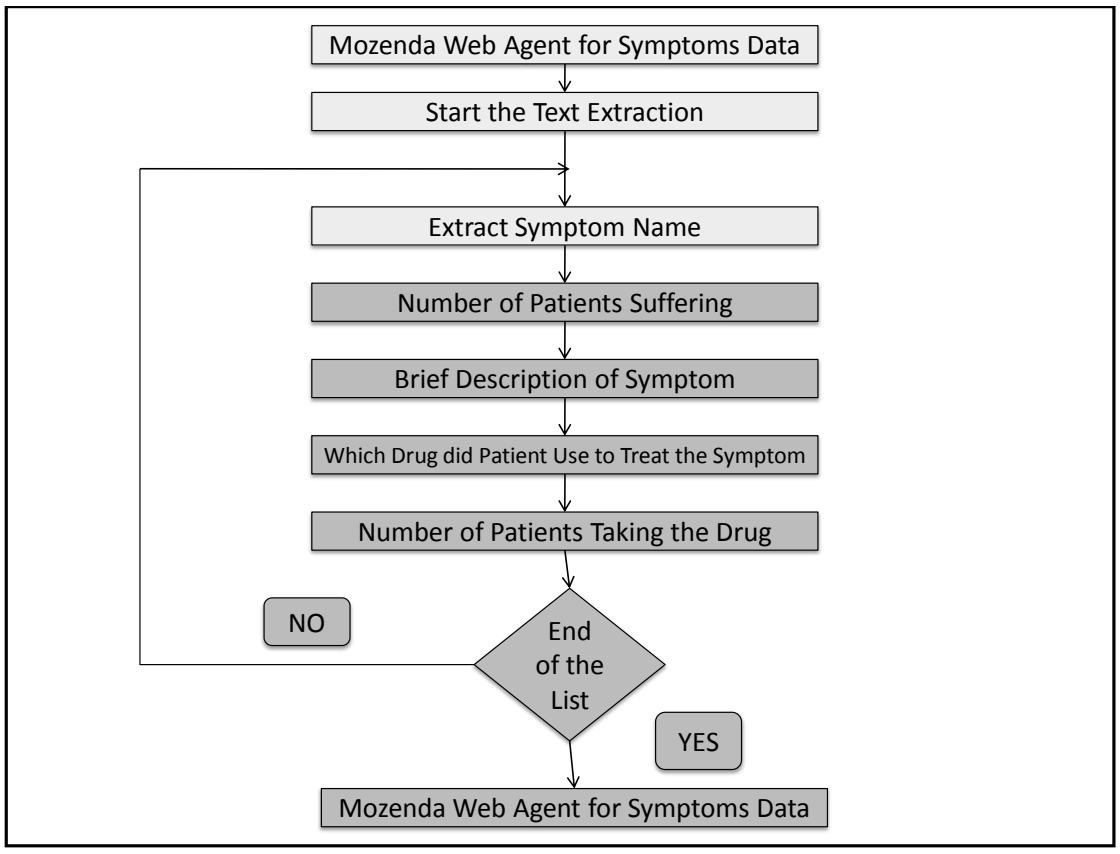

Figure 3: Algorithm for Extracting Symptoms Data from PatientsLikeMe

Example 1 below shows the Extracted Symptoms Data on Rashes.

Rashes (redness swelling),4395,A rash is an area of inflamed and irritated skin. Rashes are characterized by redness itching swelling and various types of skin lesions (e.g. macules papules nodules plaques pustules vesicles wheals). There are many different types of skin rashes.,1,Aquaphor

Rashes (redness swelling),4395,A rash is an area of inflamed and irritated skin. Rashes are characterized by redness itching swelling and various types of skin lesions (e.g. macules papules

nodules plaques pustules vesicles wheals). There are many different types of skin rashes.,1,Hydrocortisone Topical

Rashes (redness swelling),4395,A rash is an area of inflamed and irritated skin. Rashes are characterized by redness itching swelling and various types of skin lesions (e.g. macules papules nodules plaques pustules vesicles wheals). There are many different types of skin rashes.,1,Metronidazole Topical

Rashes (redness swelling),4395,A rash is an area of inflamed and irritated skin. Rashes are characterized by redness itching swelling and various types of skin lesions (e.g. macules papules nodules plaques pustules vesicles wheals). There are many different types of skin rashes.,1,Triamcinolone Acetonide in Absorbase

2) For treatments, I collected the data in the following pattern:

Drug Name, Category, Number of Patients using the Drug, Brief Description of Drug, Reasons for Taking the Drug, Number of Patients taking the Drug, Side Effects, Number of Patients having Side Effect, Side Effect Severity, Number of Patients having Side Effect Severity, Reasons for Stopping the Drug Consumption, Number of Patients that have Stopped the Consumption.

Figure 4, shows the algorithm for extracting the treatment data from the PatientsLikeMe Website. Mozenda will extract the text as per the above pattern from whole list of pages as shown in figure 4.

It will extract *Drug Name, Category, Number of Patients using the Drug, Brief Description of Drug, Reasons for Taking the Drug, Number of Patients taking the Drug, Side Effects, Number of Patients having Side Effect, Side Effect Severity, Number of Patients having Side Effect Severity, Reasons for Stopping the Drug Consumption, Number of Patients that have Stopped the Consumption,* in a single iteration then it will check for the end of the list. If the list ends, then

Mozenda will export the extracted text as a CSV file. If the list does not end, it will continue its extraction iteration.

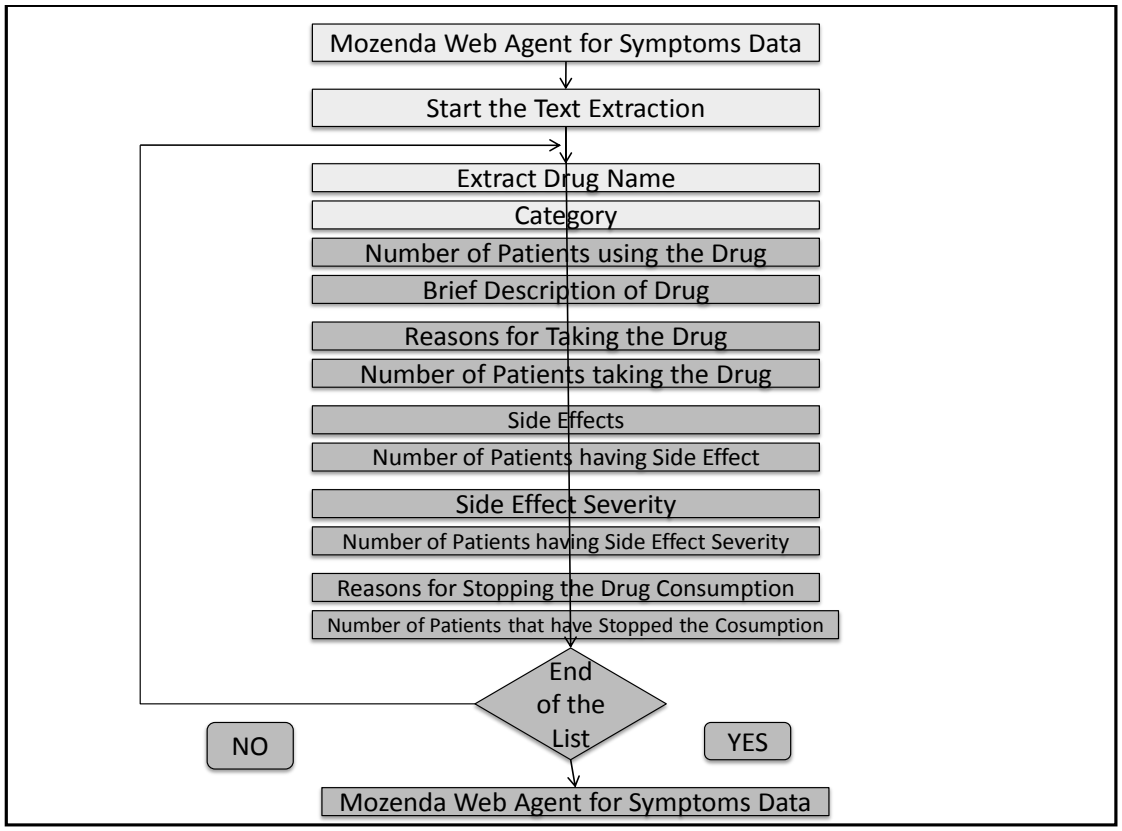

Figure 4: Algorithm for Extracting Treatments Data from PatientsLikeMe

Example 2 below shows the extracted treatments data on the drug Zytram.

Zytram XL (tramadol),Prescription Drug,6,Tramadol is an opioid analgesic used for the relief of moderate to moderately-severe pain. Extended release formulations are indicated for patients requiring around-the-clock management of moderate to moderately-severe pain for an extended period of time.,Pain,940,,,,,,

Zytram XL (tramadol),Prescription Drug,6,Tramadol is an opioid analgesic used for the relief of moderate to moderately-severe pain. Extended release formulations are indicated for patients requiring around-the-clock management of moderate to moderately-severe pain for an extended period of time.,Fibromyalgia,363,,,,,,

Zytram XL (tramadol),Prescription Drug,6,Tramadol is an opioid analgesic used for the relief of moderate to moderately-severe pain. Extended release formulations are indicated

for patients requiring around-the-clock management of moderate to moderately-severe pain for an extended period of time.,Muscle pain,79,,,,,,

Indexing

We developed an index for all the health keywords, retrieved in the data collection phase. For example, for symptom data, I built the index using the **Lucene API**. I used the Lucene API because Apache Lucene(TM) is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform. Apache Lucene is an open source project available for free download.[3]

I specified an index key as a **Symptom Name** or **Drug Name** extracted from the PatietsLikeMe Website to the Lucene API. As a result, Lucene will create an index file with CFS extension on the key. This index file is then used to find the matches in any text, i.e. the Lucene API supports a method called Search which accepts a Key (for us, drug or symptom) and the text (input text in which the matches are to be found), which returns all the matched keywords found in the input text as an output. These matched keywords are then processed further in the project to display the summary information related to the keyword.

[3] Apache Lucene API project can be downloaded from http://www.apache.org/dyn/closer.cgi/lucene/java/

4.2 Module 2: Blog Extraction Component

Figure 5 shows the detail design of blog extraction component.

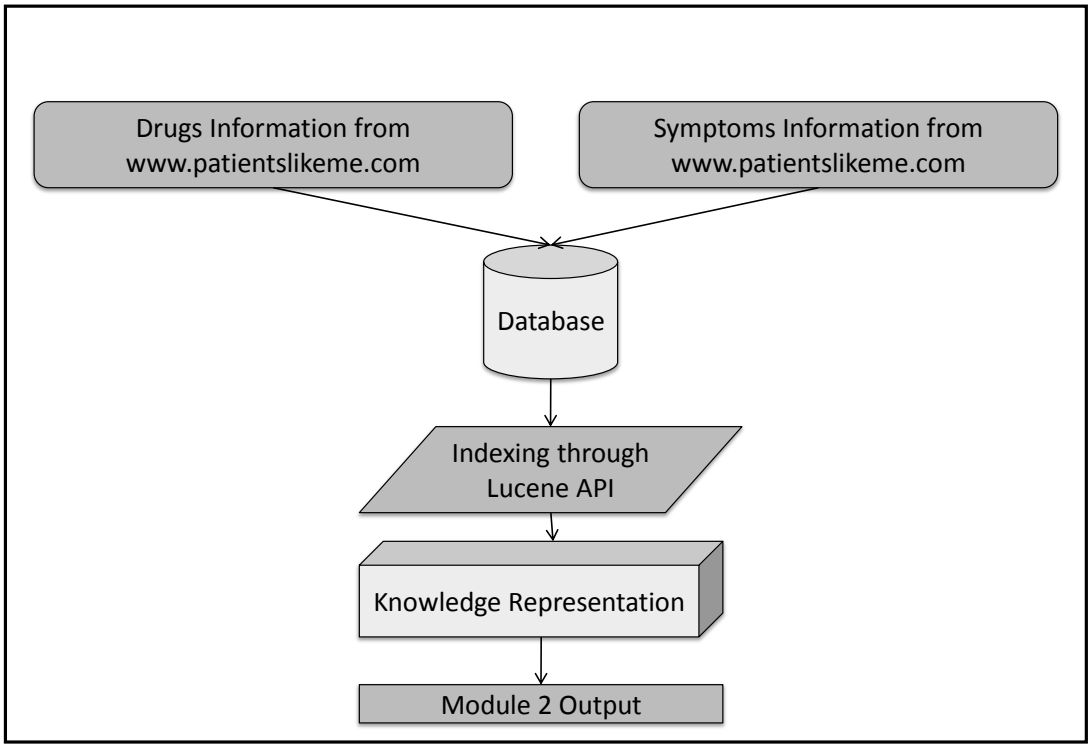

Figure 5: Detailed Design of Module 2 for Health Knowledge Refinery Portal

This module consists of Java code extracting the plain text from health blogs specified by the user in the interface. It sends the Health Blog URL to the Jsoup API, which returns the text excluding all the Meta tags and Html tags, leaving just the plain text.

The social media-based health discussions and blogs in the examples below illustrate the data characteristics.

Example 1 Breast Cancer Data[4]

I had Inflammatory Breast Cancer. Even four years afterward part of me still doesn't feel free of it. I guess most of me has moved on, though.
One in eight women will get breast cancer. That's roughly 12.5 women out of every 100. Only 5 percent of all those women with breast cancer will have IBC. It's an exclusive club you don't want to belong to… Yet I am becoming more and more aware of repeating patterns of Inflammatory Breast Cancer association.
When I was first worrying about all the changes happening in my breast and beginning the cluster of appointments that would lead to my diagnosis, my new boy friend told me that I had a Get out of Cancer Free card. Her previous friend had died of breast cancer just a few years earlier and she believed "God wouldn't do that to her twice." Heh. Of course, I was diagnosed despite her assurances. We spent many long phone conversations rehashing her friend's symptoms, illness and death. We came to believe that she had most likely also had IBC.
In August I met Kelly soliciting donations in front of Wal Mart for her Avon Walk for Breast Cancer. She was walking Santa Barbara. I gave her a small donation and she handed me a pamphlet on IBC. I was more than a little surprised. Not many know about Inflammatory Breast Cancer. "Who had IBC?" I asked her. Turns out her mother had died in only 18 months after her diagnosis. When she found out I'd had IBC she seemed stunned. Her voice quieted as she told me that she'd never met an Inflammatory Breast Cancer survivor before.
Wow. It was a poignant moment for me. I still remember quite clearly finding the Survivor Stories online. They were so very encouraging… but also very old. At the time I don't think there was a post newer than 3 years old. I couldn't help but wonder if any of those women were still alive. Kelly's admission gave me a rush of emotions. Pride – that I'd beaten the beast and could stand there to tell about it. Sadness – that there are so few IBC survivors. And happiness that I was able to raise a living, active voice above the silence of lost women.
Ironically, Kelly knew her mother had IBC because her bff's mother had also had it. Am I the only one that wonders at the mysteries and coincidences of the universe?
Fast forward to this week when Kelly & I helped our local morning news anchor with a story on IBC. Here's the video if you're interested. Please listen to the symptoms. Most of all, tell the other women in your life about them, too. Kelly knew the symptoms from listening to her friend talk about her mother, though she's not sure if she ever shared them with her own mother. I knew something was wrong before diagnosis but delayed talking to anyone about it until it was almost too late.
I'm one of the lucky ones.

[4] This data was collected from http://motherswithcancer.wordpress.com/

Example 2 Breast Cancer Blog Data[5]

I live in Upstate New York and was diagnosed with IBC on March 11, 2009, at age 42. I was still nursing my baby. In January I had a clean phyisical (with breast exam) and I first noticed something in my breast was different on Feb. 21.
I was diagnosed in Syracuse, NY. Staging at my local hospital (CT and bone scan) didn't notice mets. Since my brother knew an oncologist at MD Anderson in Houston, I went there and had a PETscan which confirmed that I was Stage IV, with mets to my spine, hip/pelvis, and one to my liver. The "tumor" was ca. 7x9 cm in my right breast, about 2/3-3/4 of my breast was hard. ER+, PR-, HER2+.
I received 6 rounds of chemo, starting March 30 and ending July 13, 2009. My chemo was TCH (Taxotere, Carboplatin, Herceptin) and I also received Zometa. I went from breastfeeding to menopause in a very short time.
By the end of my chemo, they could no longer detect any cancer on the scans (I went to MD Anderson's IBC clinic before, midway and at the end of my chemotherapy). While Dr. Cristofanilli recommended surgery followed by radiation at MD Anderson, I decided to go with another physician, Dr. Maria Theodoulou, at Memorial Sloan-Kettering in New York City, much closer to my home. She and surgeon Monica Morrow recommended waiting 6 months after my chemo ended, and if the cancer continued to stay in check, we would look at surgery (rad. mod. mastectomy) followed by radiation.
I had a scan at the end of October, everything still looked good. Since my chemo ended, I continue to receive Herceptin every 3 weeks, and get a shot of Lupron (to keep me in menopause) and Zometa every 12 weeks. I also take Femara daily.
It seems counter-intuitive to many to have surgery when they cannot see any cancer, but the physicians say it is just too small to see, not gone -- they cells are perhaps dormant, and waiting to make their comeback. They feel that surgery and radiation may help, though they cannot really prove this. I feel lucky, things have gone as well as they could have since my diagnosis. I am feeling well (true, I only felt unwell from the chemo, and it has taken a bit to rebound) and feel I have grown in many ways from what I've been through.
I have scans coming up on Feb. 1st, and a decision about surgery/radiation/reconstruction Feb. 2nd. I am nervous about surgery and radiation, not exactly looking forward to it. One of the physicians I saw for a second opinion last June, Dr. Linda Vahdat at Cornell-Weill Medical Center in New York, said she wouldn't do surgery, she would just treat with drugs. Another physician, Dr. Beth Overmoyer at Dana-Farber in Boston, said she would basically do the same as MD Anderson (except she would also do an oopherectemy, while Dr. C at MD Anderson said "no one does those anymore"). Any and all thoughts and suggestions are welcomed!

After reading these two examples, we can conclude that the textual information in medical blogs is very verbose and unstructured, and important information or points are fragmented in different

[5] This data was collected from http://www.ibcsupport.org/list/2010-01/0587.html

parts of discourse. It would be time consuming to read all the conversations in order to gather the needed information. So we developed an integrated system that has most of the medical vocabulary stored as an index that helps us to find the matched keyword in the text, as above. These matched keywords are highlighted in the plain text. We also add WebMD and PubMed links for each matched keyword. The user can see the brief description and related statistics for each keyword by clicking on it. The major steps accomplishing this task are as follows.

4.2.1 Health Blog Data Extraction

Figure 6, shows the entry screen of **Health Knowledge Refinery Portal.**

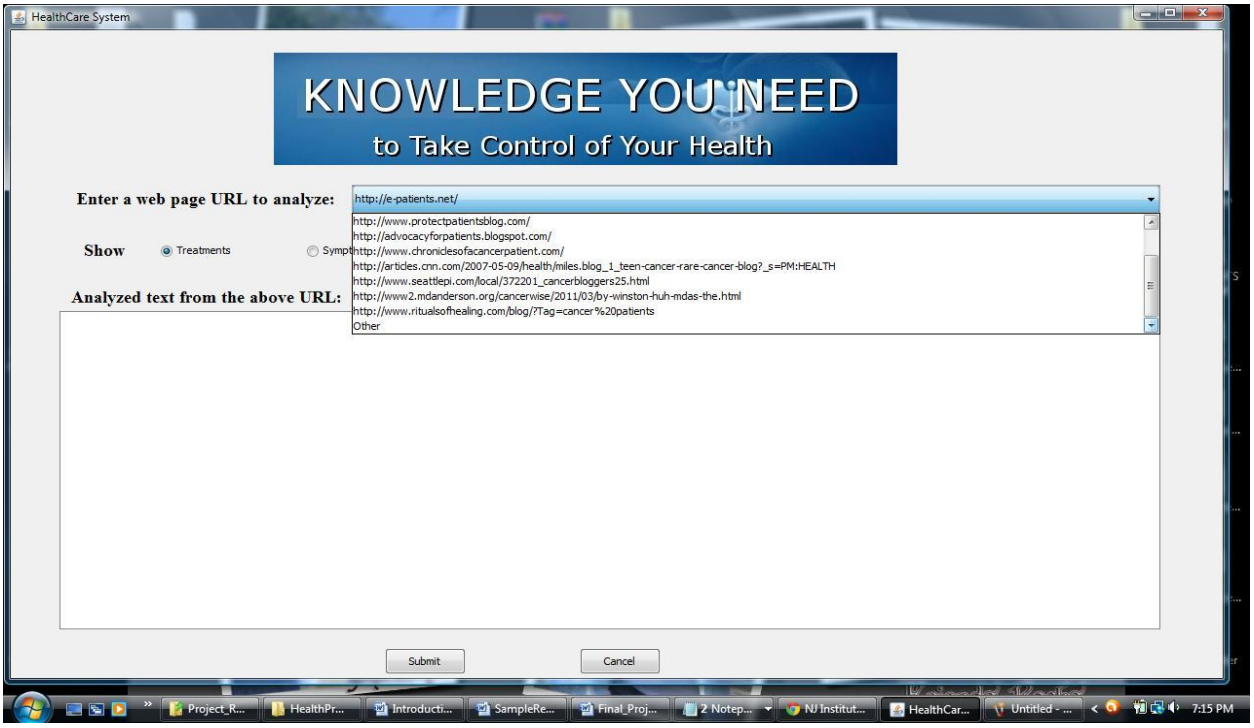

Figure 6: List of Predefined URLs for Health Blog Pages

Users have to select a URL from a predefined list provided in a project or they can specify a new URL. The URL that is needed as an Input for this project must be a valid Health Blog URL. The URL may also point to a Website that contains health discussions, personal stories submitted by

users, etc. This URL is then provided to the ***jsoup* API**. The **CANOLA EXTRACTOR**[6]**,** a full-text extractor method of ***jsoup*** is used to extract the plain text from the URL target. This method removes all the Meta tags, Html tags, and CSS tags from the URL and returns only the plain text.

For extracting the text with the **CANOLA EXTRACTOR** method of ***jsoup* API,** I wrote code to access the ***jsoup* API** specifying the method to extract the text as a **CANOLA EXTRACTOR.** This method needs a valid input and the desired output format specified by the user; so I gave a URL (submitted by the user) as input parameter and the output format as "plain text." Figure 7, shows the detailed design for extracting the plain text from health blog URL.

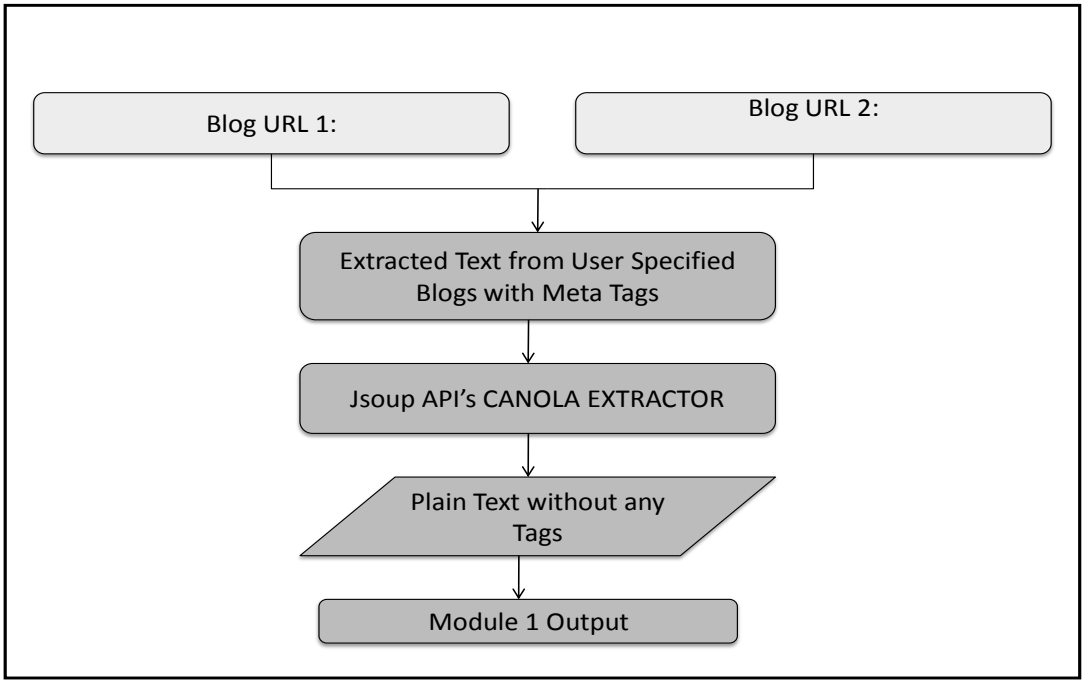

Figure 7: Detailed Design of Blog Extraction Module

[6] A full-text extractor method of JSOUP API

4.3 Module 3: Searching for Health Keyword Identification

After validating the URL, users are asked to select the search criteria, i.e. either search for symptoms or search for treatments in the extracted text from the specified URL. Figure 8, shows the selection of search criteria to analyze.

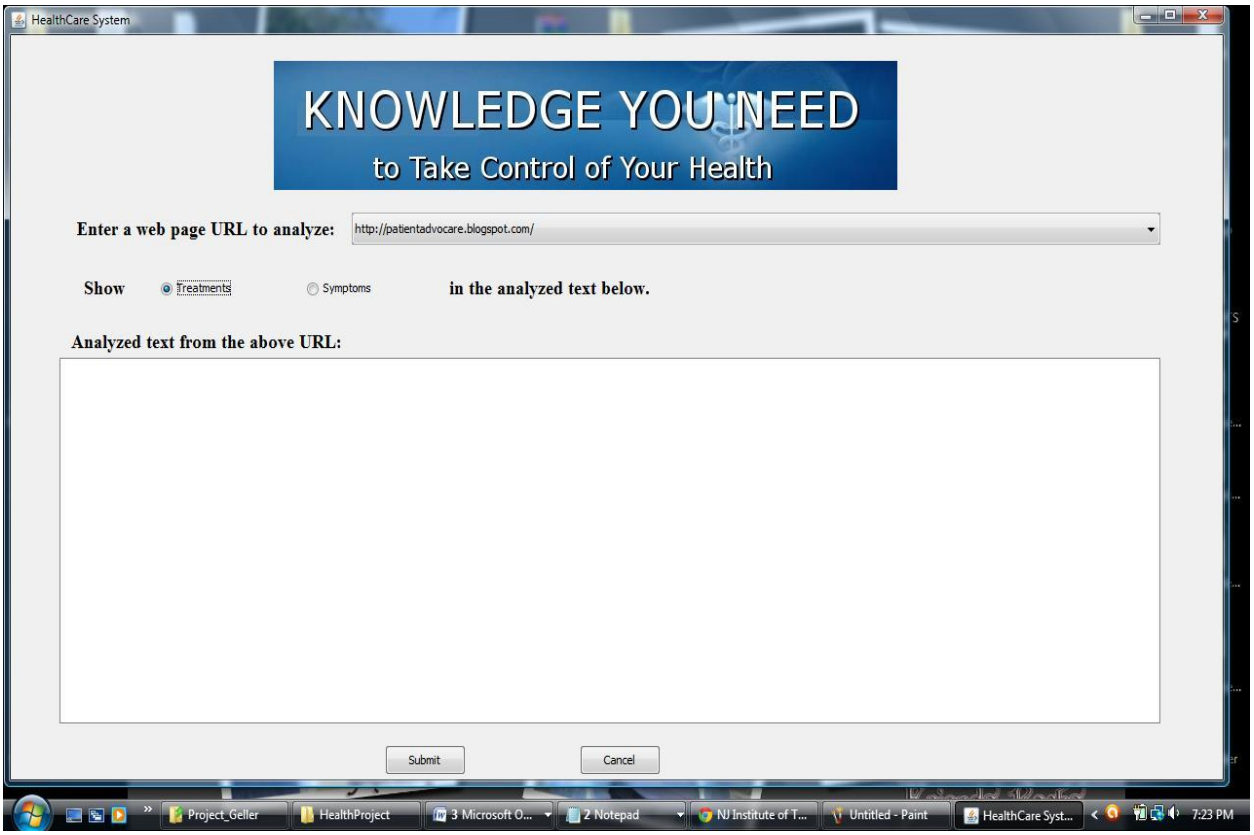

Figure 8: Selecting Treatments as a Search Criteria

If the user wants to retrieve all the symptoms from the specified health blog URL, he may select the symptom radio button and then submit the query. The Health Knowledge Refinery Portal will then display the text retrieved from the specified health blog URL, along with all the matched symptoms highlighted. The same is true for treatments or drugs.

4.3.1 Searching the Text for Symptoms

I used the index built on symptoms data for searching the symptoms in the extracted text. The index on symptoms data is built on the symptom name, so it is easy to find the word in the extracted plain text that matches the word in the index file. Such a matched word is then highlighted in the extracted text. Figure 9, shows the result of selecting symptom as search criteria to analyze.

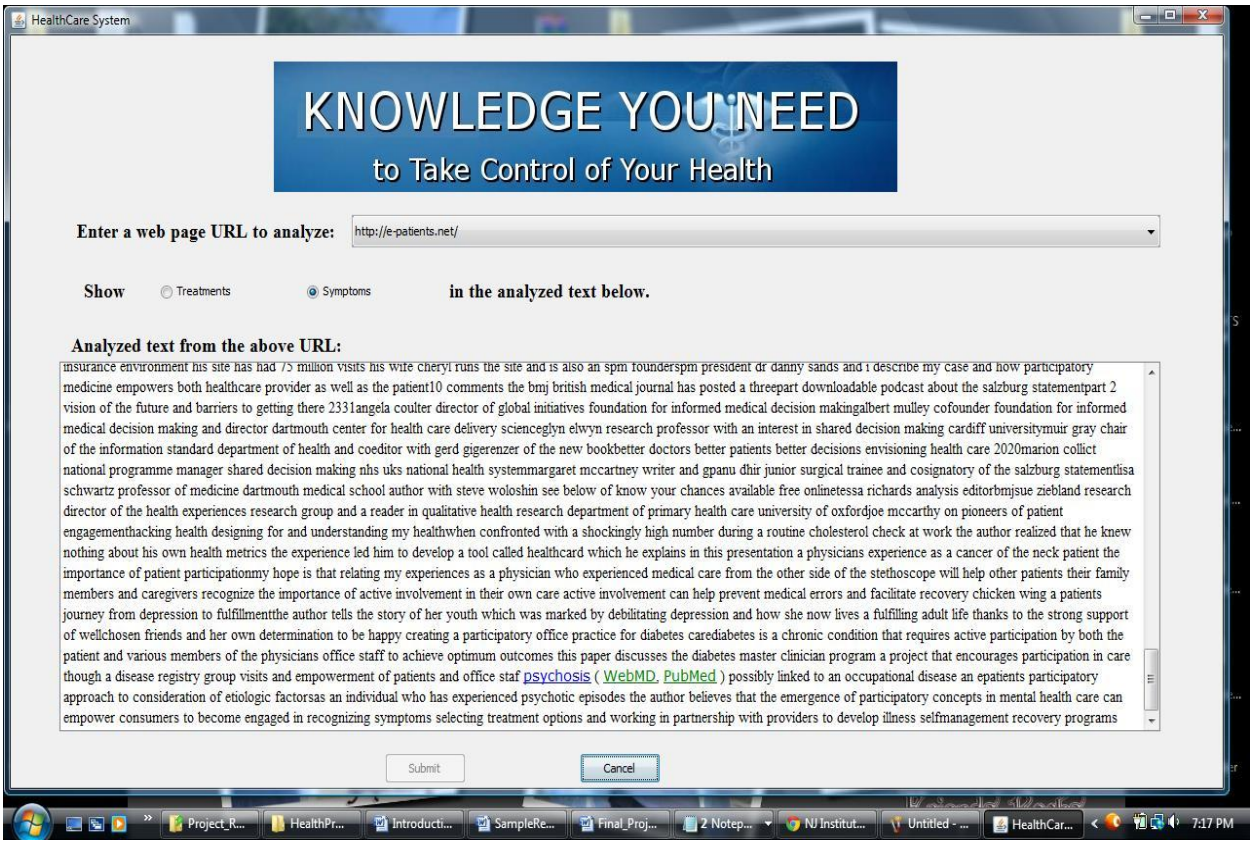

Figure 9: Result of Symptoms Search Criteria

Each of these matched symptoms has a hyperlink associated with it. After clicking on the matched symptom, all the brief details of the selected symptom are displayed along with the adequate statistics in tabular format. I also provide WebMD and PubMed links for all the

matched symptoms. When any user clicks on the WebMD or PubMed link beside the matched symptom, the relevant WebMD or PubMed page is displayed to the user in the default Web browser.

4.3.2 Searching the Text for Treatments/Drugs

Figure 10, shows the result of selecting treatment as search criteria to analyze.

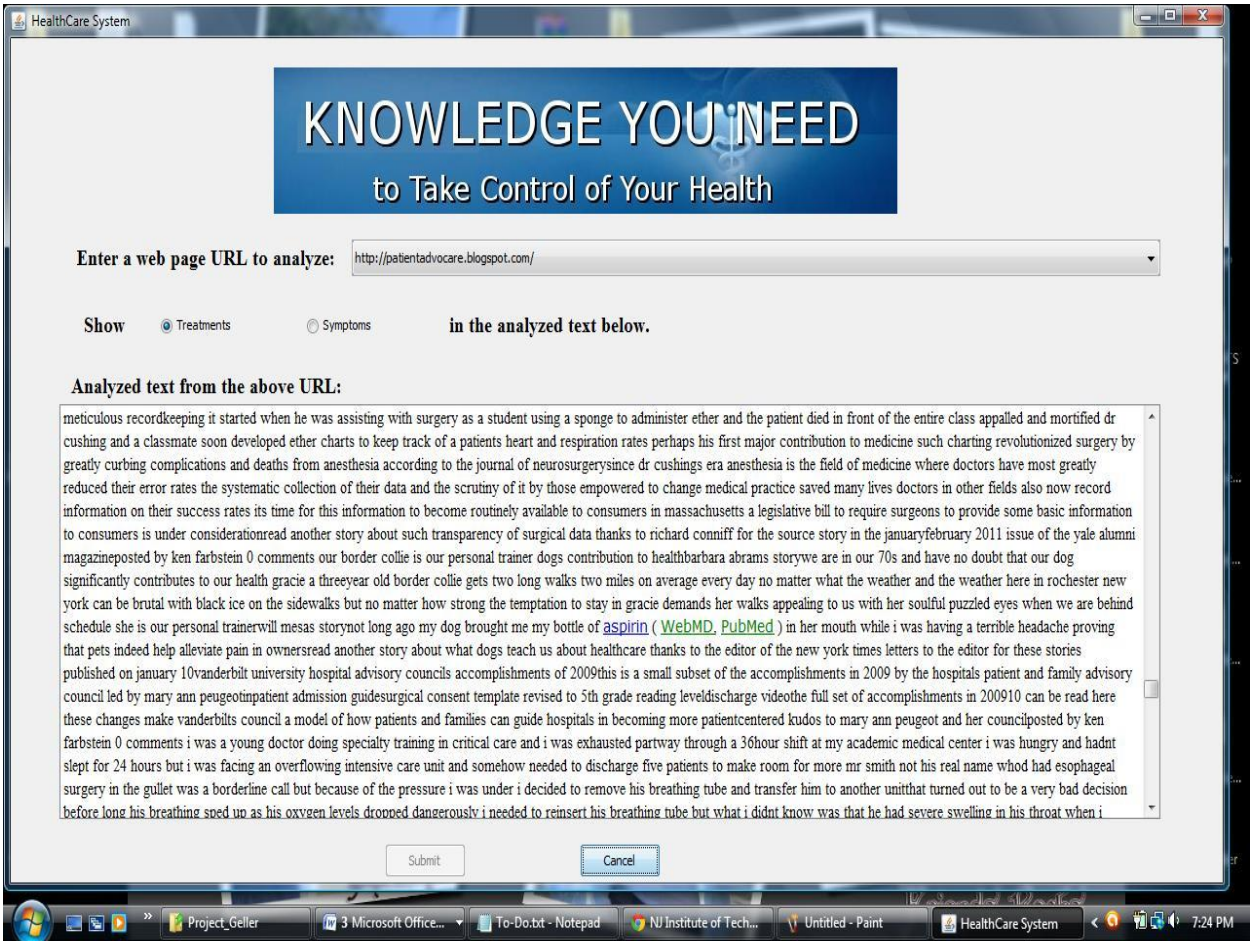

Figure 10: Result of Treatments Search Criteria

I used the index built on treatments data for searching the treatments in the extracted text. The index on treatment data is built on the treatment name, so it is easy to find the word in the

extracted plain text that matches the word in the index file. Such a matched word is then highlighted in the extracted text. Each of these matched treatments/drugs has a hyperlink associated with it, just as for the case of symptoms.

4.3.3 Highlighting Keywords and adding Hyperlinks

Figure 11, shows the highlighted symptoms and external links as a result of selecting symptom as search criteria to analyze.

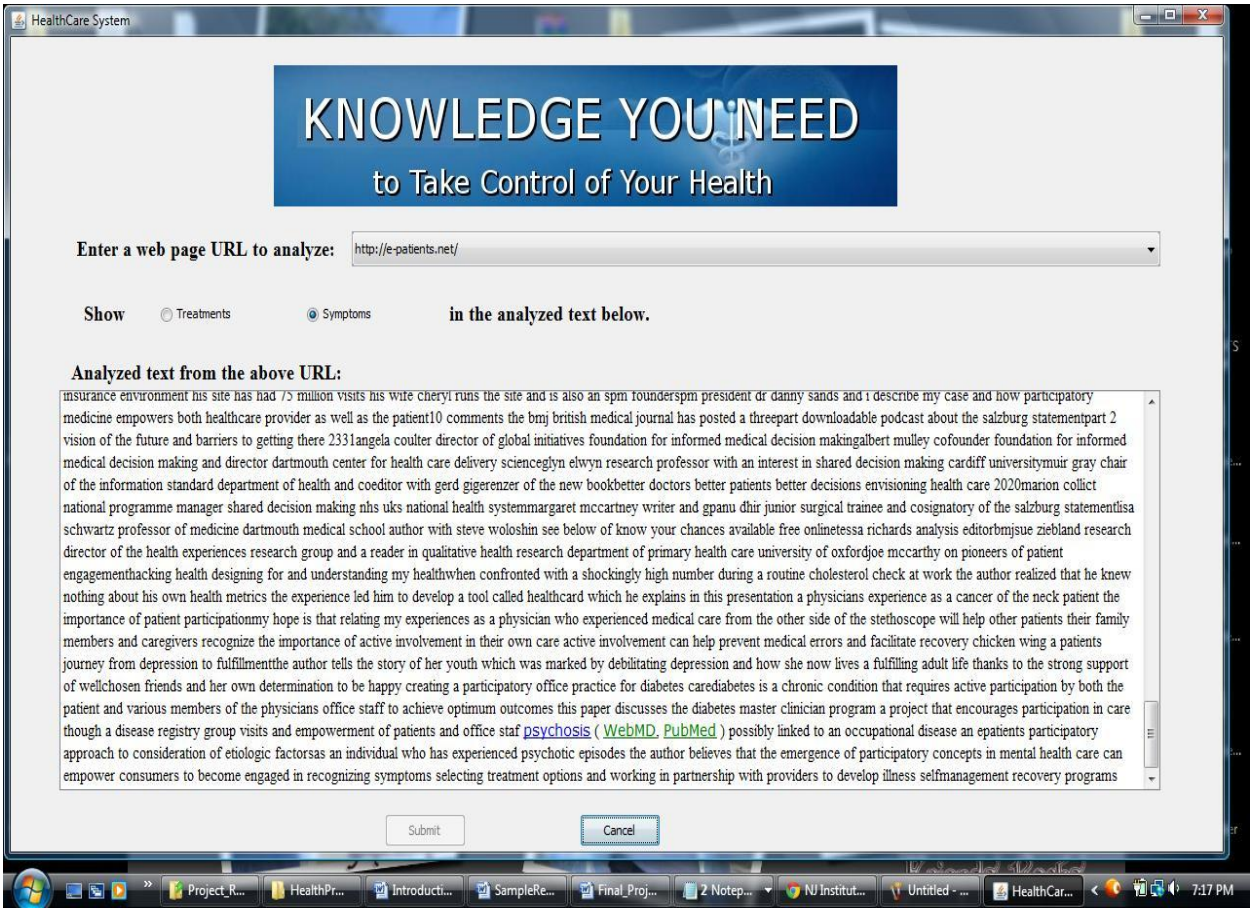

Figure 11: Highlighted the Matched Keyword *Psychosis* and Added External Links

As per search criteria, either all the matched symptoms or all treatments/drugs are highlighted in the extracted text. The whole text along with these highlighted keywords is displayed to the

users. I have associated a hyperlink with all of these matched keywords so that users can click on any of this matched keyword to get the brief description of that keyword along with the statistical tabular data.

4.4 Module 4: Integrated Health Summary and Linking Component

Figure 9, shows the detail design of integrating and linking **Health Knowledge Refinery Portal**

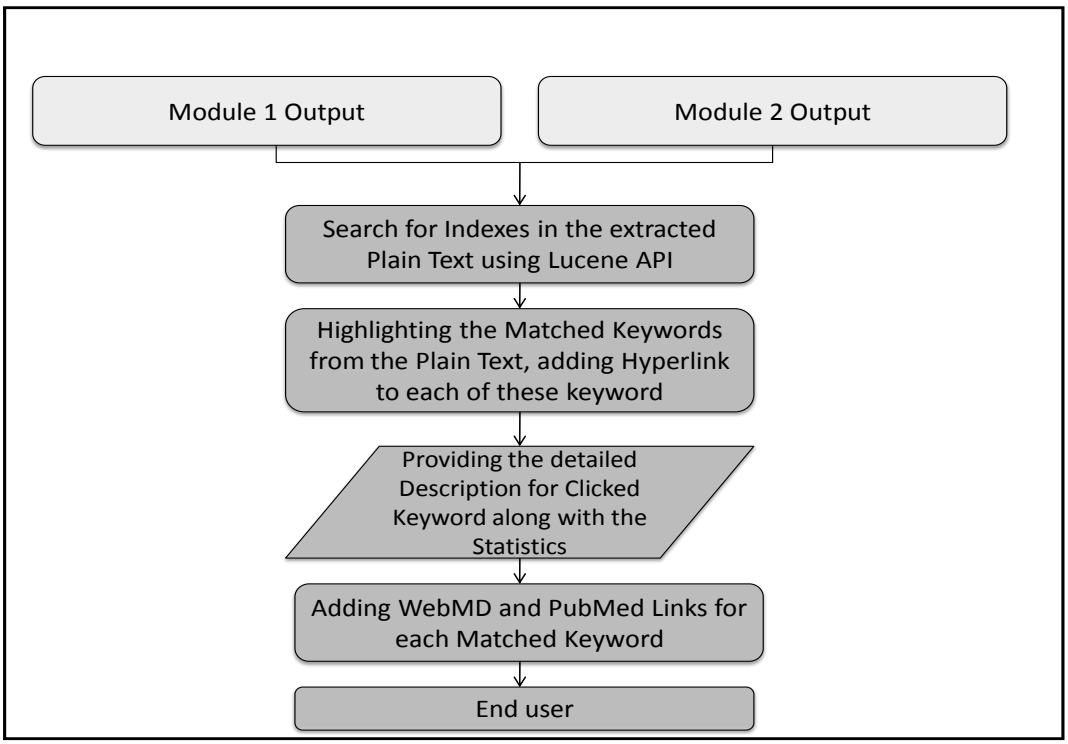

Figure 12: Detailed Design of Module 3 for Health Knowledge Refinery Portal

Module 3 consists of Java code integrating the output of Module 1 and Module 2 and producing the highlighted text. The indexes created in Module 2 are used to match the keywords from the plain text extracted from Module 1 using the Lucene API. As the matches are found, the program highlights the matched keyword and adds a hyperlink to each. When a matched keyword is

clicked, the brief summary of the keyword along with the adequate statistics are displayed to the user in a JOptionPane. We also provide WebMD and PubMed links for all the matched keywords.

4.4.1 Calculating the Statistics

Figure 13, shows the result of clicking the highlighted keyword ***psychosis***.

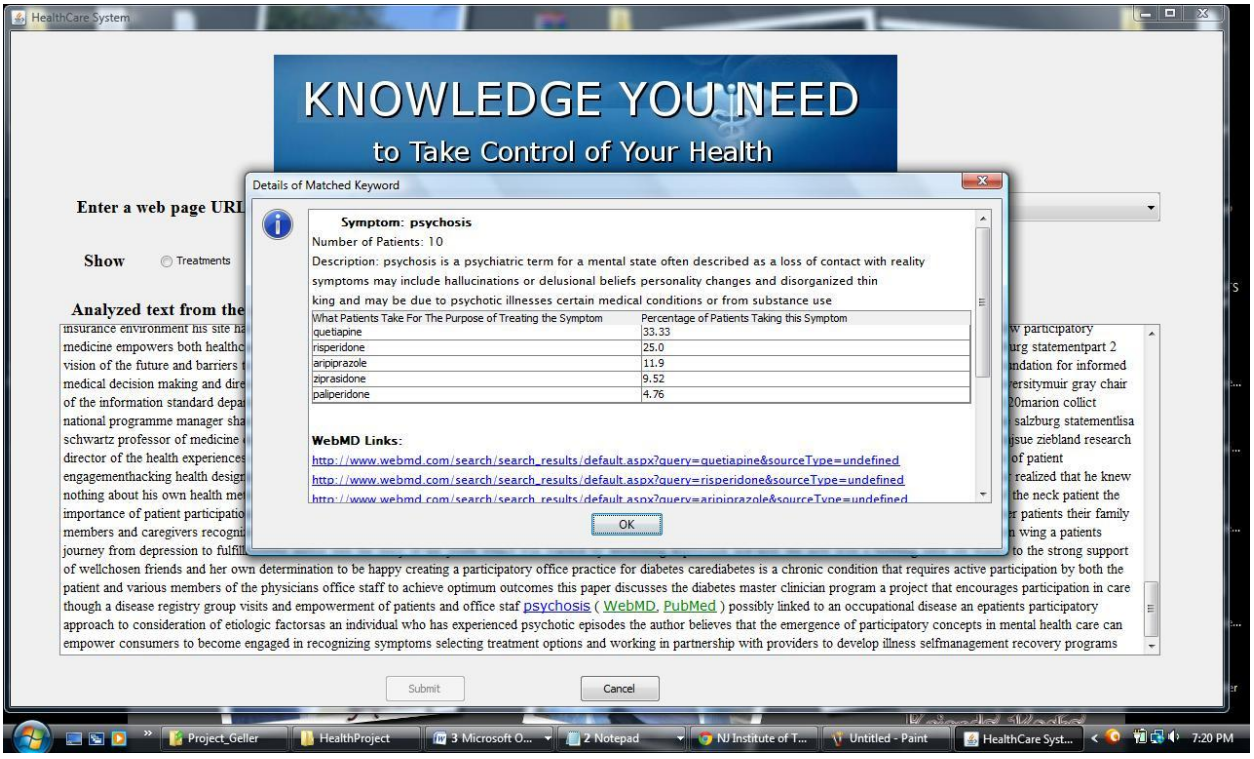

Figure 13: Result of Clicking on the Matched Keyword *Psychosis*

Data collected from www.patientslikeme.com does not have any statistical data associated, they only provide the raw numbers. So, I calculated the statistics from the collected data. Say for symptoms, I calculated the statistics as percentage of patients that take the particular drug for

treating the respective symptom. I displayed only the top five drug names along with their respective percentages.

For each matched keyword, I retrieved name and the type of the keyword, i.e., whether the keyword is symptom or drug. If the keyword is "drug" then the category of the drug, total number of patients related to that keyword, a brief description about the keyword and the tabular statistics are displayed.

For calculating the statistics for Symptoms, I stored a list of the number of patients taking the particular drug for treating the matched symptom in a Hashmap as a Key - Value pair, where the drug that is used for treating the symptom is a Key and the respective number of patients taking this drug is the Value. After storing the complete list as Key – Value pairs, I calculated the sum of the total number of patients stored in a Hashmap. This number is then used to divide each individual number of patients in the Hashmap, one by one, and is finally multiplied by 100 to calculate the percentage of patients taking the particular drug for treating the above matched symptom. I displayed only the top five drug names sorted in descending order of calculated percentage.

I also constructed a customized URL for WebMD and PubMed, for each of these five drug names, so that if a user wants to get some detailed information about any drug specified, they can easily be routed to the appropriate destination.

For calculating the statistics for treatment, I stored a list of reasons for taking this particular drug in a Hashmap as a Key - Value pair, where the reason for taking this drug is a Key and the respective number of patients is the Value. After storing the complete list as a Key – Value pair,

I calculated the sum of total number of patients stored in a Hashmap. This number is then used to divide each individual number of patients in a Hashmap one by one and is finally multiplied by 100 to calculate the percentage of patients. I displayed only the top five reasons sorted in descending order of calculated Percentage.

I also constructed a customized URL for WebMD and PubMed, for each of these five drug names, so that if a user wants to get some detailed information about any drug specified, they can easily be routed to the appropriate destination.

This process is repeated for calculating the statistics for Side Effects of this Drug, Side Effect Severity of this Drug, and Reasons for stopping this Drug Consumption.

4.4.2 Adding External Links for Expert Knowledge and Research Results

Figure 14, shows the result of clicking WebMD link next to ***psychosis***.

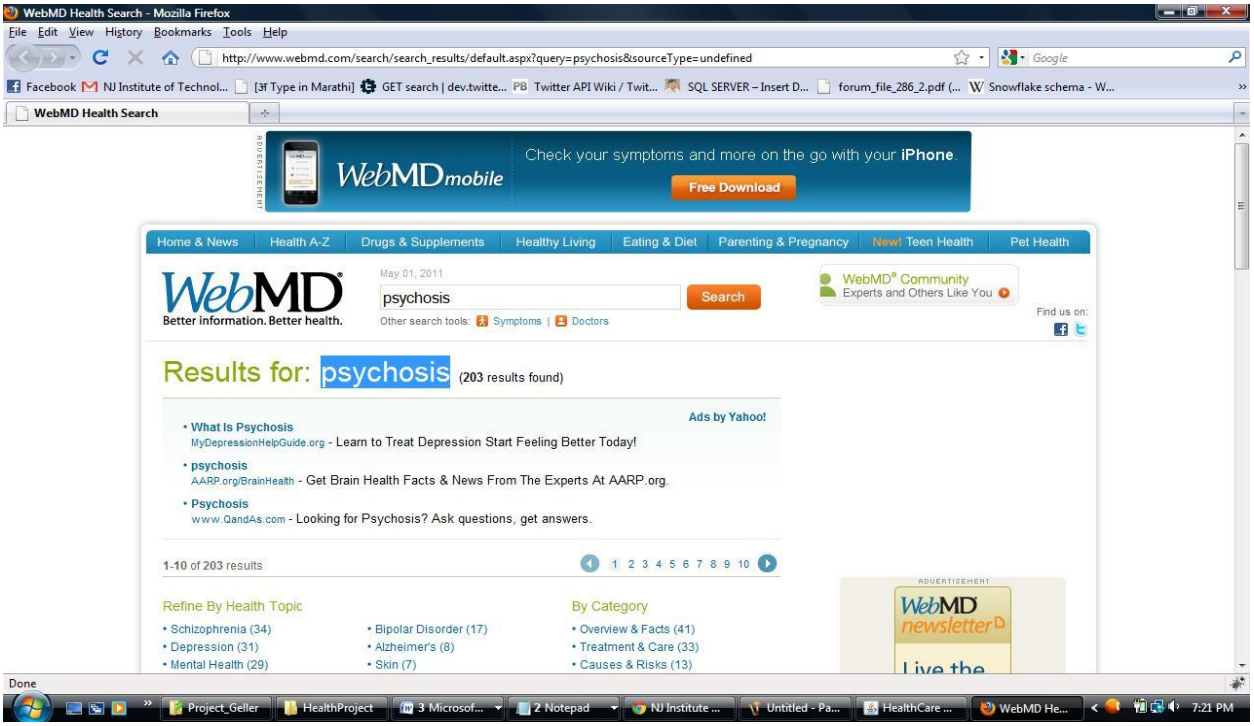

Figure 14: Result of Clicking on the WebMD link besides *Psychosis*

Figure 15, shows the result of clicking PubMed link next to ***psychosis***.

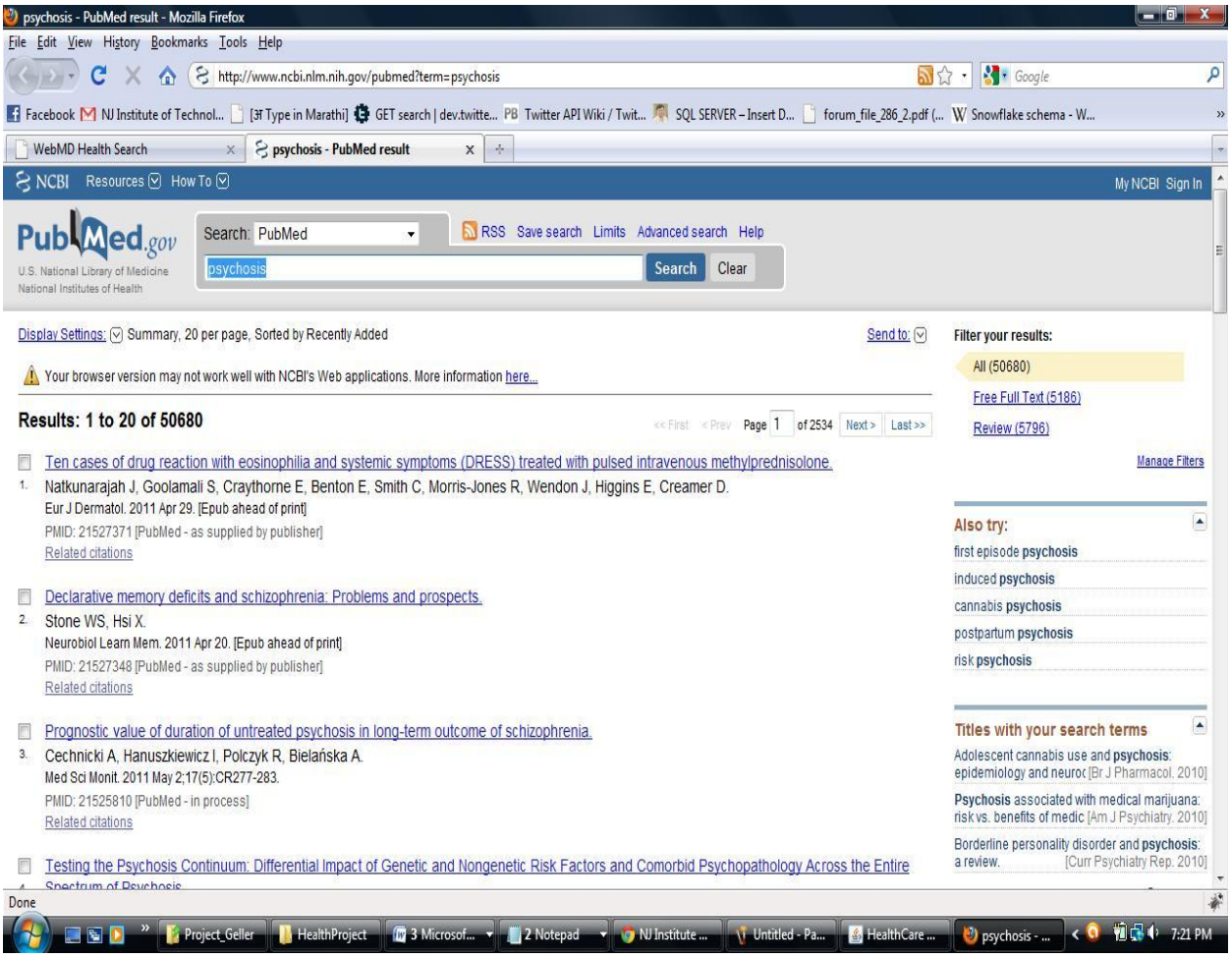

Figure 15: Result of Clicking on the PubMed link besides *Psychosis*

I gave the WebMD and PubMed links for each matched keyword, so that every time the user clicks on a WebMD link next to the highlighted keyword, the default browser displays the WebMD page for the matched keyword so that the users can collect more information about that. The same is done for Pubmed links.

5. Conclusions

Useful information for health care purposes can be acquired through the Internet, but most of the times this information is not so concise, or only a part of the whole document is related to our interests. Health knowledge is widely distributed over the Internet. Users have to make extra efforts to gain the detailed knowledge for a particular drug or symptom. They have to visit many different Websites, for example www.patientslikeme.com, www.Webmd.com, www.pubmed.com or read a whole document or page to gather the information about a particular drug or symptom. This information neither exists in a summarized manner nor does the information provide any statistics.

Also, all health information is sensitive. A recommendation for a drug may create adverse effects if the treatment goes wrong. Therefore, there is a definite need of a system that integrates comprehensive information about the drugs and symptoms from several different sources, such as those mentioned above, calculate useful statistics and display all this in a single independent system. Thus, we developed the **Health Knowledge Refinery Portal**, which produces a summary of the problems and symptoms faced by many patients and the possible solutions or treatments. This helps the users with understanding the trends in health care and make more informed decisions with integrated knowledge for their health care purposes. Any user, whether patient or doctor or health care staff member can acquire knowledge of symptoms and possible treatments in a summarized manner and with less effort and time.

The **Health Knowledge Refinery Portal** we developed consists of the following components:

1) Social Health Knowledge Base: This component uses Web data extraction and data mining to extract the rich social health knowledge on treatments, symptoms, and side effects, contributed

by patients, and construct a health knowledge base from it. This social health knowledge is valuable since it is based on a collection of actual patients' experiences.

2) Blog Extraction Component: To illustrate how to support the users with the social health knowledge base extracted from social health networks, we developed a component to identify health terms in a user input health blog. This blog extraction component extracts all the text from the blog URL specified by the user using the ***jsoup*** API

3) Health Keywords Identification component: Once the text is extracted from a designated blog, the Health Keyword Identification component looks for the health-related keywords that matched with the keywords (drugs or symptoms) stored in the Social Health Knowledge Base.

4) Integrated Health Summary and Linking component: All the matched heath-related keywords i.e. either ***Drugs*** or ***Symptoms*** are marked up with health summary data from the Social Health Knowledge Base and with hyperlinks to WebMD and PubMed medical knowledge relevant to the health keywords, providing an integrated health and medical information environment.

How to use Health Knowledge Refinery Portal

Any user that wants to access the system has to provide a valid URL. Users can select a URL from a predefined list of health discussion URLs or they can enter a URL that they want to analyze. Users may select "drugs" or "treatments" as search criteria and as a result get all the available relevant information in a summarized manner. This information may contain the past experience of patients and effects of drugs/treatments observed by them.

Users can click on any keyword to get summarized information of that keyword from our knowledge base. The summary for any matched keyword consists of a ***detailed description*** of the keyword and ***statistics*** featuring ***percentages*** of patients taking a particular drug, ***side effects*** of the drug, ***severities of the side effects***, ***reasons for stopping*** the drug consumption, etc.

Percentages of patients taking a particular drug for a common symptom constitute one of the important factors to decide what would be the best drug among the list for treating a specific symptom. Side effects of consuming the particular drug along with the statistics and severities of the side effects help us to get adequate knowledge about that particular drug. Reasons for stopping the use of a particular drug helps users know whether they should consider stopping the drug consumption.

References

1. Tim Weninger, William H. Hsu, Text Extraction from the Web via Text-to-Tag Ratio, Database and Expert Systems Application, 2008. DEXA '08. 19th International Workshop, 2008.
2. Kerstin Denecke, Studying Content Differences of Medical Online Web Resources, 2008.
3. Saiful Akbar, Laura Slaughter, Oystein Nytro, Collecting Health Related Text From Patient Health Writings, Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference, 28 February, 2010.
4. Maaike van den Haak, Charlotte van Hooijdonk Evaluating Consumer Health Information Websites: The Importance of Collecting Observational, User-driven Data, Professional Communication Conference (IPCC), 2010 IEEE International, 7 July, 2010.
5. Mostafa M. Aref Zhengbo Zhou, The Ontology Web Language (OWL) for a Multi-Agent Understating System, Integration of Knowledge Intensive Multi-Agent Systems, 2005. International Conference, 18 April, 2005.
6. Silvia Baptista, Integrating Medical Text Extraction Tools, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, January 31, 2008.
7. Web Data Extraction Software Tool, http://www.mozenda.com/Web-data-extraction, Accessed, April, 2011.
8. Aetna InteliHealth – The Trusted Source, http://www.intelihealth.com, Accessed, February, 2011.
9. Patients Like Me, Accessed http://www.patientslikeme.com, Accessed, April, 2011.

10. Web MD – Better Information. Better Health,

http://www.Webmd.com, Accessed, May 2011.

11. Cancer Blog,

http://cancerblog.blogspot.com/, Accessed, April 2011.

12. Inflammatory Breast Cancer Mailing List By Thread,

http://www.ibcsupport.org/, Accessed, April, 2011.

13. Mothers With Cancer – Raising Children, Fighting Cancer Living Life,

http://motherswithcancer.wordpress.com/, Accessed, April, 2011.

APPENDIX A: USER MANUAL

A.1 Initial Setup and Software Required

- If SUN's JDK is not installed on your system, Download and install the latest version of Sun JDK from http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u25-download-346242.html

A.2 Building the Health Knowledge Refinery Portal

- UnZip (Extract all the files from) the 'HealthProject.zip' folder anywhere on your System.

A.3 Executing the current build of the Health Knowledge Refinery Portal

- *Double-Click* the *Health_Knowledge_Portal.jar* file to run the application.
- Select the URL from a pre-defined list of URL or specify a new URL, select any of Treatments or Symptoms button and press Submit button.
- You will have to wait for maximum one minute, to get the output.

APPENDIX B: CODE

Health.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Map;
import java.util.TreeMap;
import javax.swing.*;
import javax.swing.event.HyperlinkEvent;
import javax.swing.event.HyperlinkListener;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.TableColumn;
import javax.swing.text.BadLocationException;
import javax.swing.text.Style;
import javax.swing.text.StyleConstants;
import javax.swing.text.StyledDocument;
import javax.swing.text.html.HTMLEditorKit;
import org.apache.lucene.index.IndexWriter;
import org.xml.sax.InputSource;
import de.l3s.boilerpipe.document.TextDocument;
import de.l3s.boilerpipe.extractors.CanolaExtractor;
import de.l3s.boilerpipe.sax.BoilerpipeSAXInput;

public class Health extends JFrame {
        // Variables declaration
        private JLabel jLabel1;
        private JLabel jLabel2;
        private JLabel jLabel3;
        private JTextField jTextField1;
        private JTextPane jTextArea1;
        private JScrollPane jScrollPane1;
        private JButton jButton1;
        private JButton jButton2;

        private JLabel jLabel4;

        private JRadioButton jCheckBox1;
        private JRadioButton jCheckBox2;

        private ButtonGroup bgroup;

        private JLabel jLabel5;

        Boolean check1 = false;
        Boolean check2 = false;

        HashMap hash = new HashMap();
        static HashMap hash2 = new HashMap();
        static HashMap hash3 = new HashMap();
```

```
HashMap show_symp = new HashMap();
HashMap show_treat = new HashMap();

HashMap linked = new HashMap();
HashMap linked2 = new HashMap();

int columncount = 0;

ArrayList<String> columntext = new ArrayList<String>();

ArrayList<String> columntext45 = new ArrayList<String>();
ArrayList<String> columntext67 = new ArrayList<String>();
ArrayList<String> columntext89 = new ArrayList<String>();
ArrayList<String> columntext1011 = new ArrayList<String>();

HashMap rowval = new HashMap();
ArrayList<String> rowceil = new ArrayList<String>();
HashMap ceilval = new HashMap();

HashMap rowval1 = new HashMap();
ArrayList<String> rowceil1 = new ArrayList<String>();
HashMap ceilval1 = new HashMap();

HashMap rowval2 = new HashMap();
ArrayList<String> rowceil2 = new ArrayList<String>();
HashMap ceilval2 = new HashMap();

HashMap rowval3 = new HashMap();
ArrayList<String> rowceil3 = new ArrayList<String>();
HashMap ceilval3 = new HashMap();

static int ccount = 0;

String[] rowtext = {};

JTextPane textPane;
JTextPane textArea;

int er = 0;

ArrayList<Integer> myintArr = new ArrayList<Integer>();
ArrayList<Integer> myintArr2 = new ArrayList<Integer>();
String check;

float t11 = 0, t2;
float t1check1 = 0, t2check1;

private JPanel contentPane;
URL url;

// End of variables declaration

public Health() {
        super();
        initializeComponent();
```

```
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setVisible(true);
    }

    private void initializeComponent() {
            jLabel1 = new JLabel();
            jLabel2 = new JLabel();
            jLabel3 = new JLabel();
            jTextField1 = new JTextField();
            jTextArea1 = new JTextPane();
            jScrollPane1 = new JScrollPane();
            jButton1 = new JButton();
            jButton2 = new JButton();

            jLabel4 = new JLabel();
            jLabel5 = new JLabel();

            jCheckBox1 = new JRadioButton();
            jCheckBox2 = new JRadioButton();

            bgroup = new ButtonGroup();
            bgroup.add(jCheckBox1);
            bgroup.add(jCheckBox2);

            contentPane = (JPanel) this.getContentPane();

            //
            // jLabel1
            //
            jLabel1.setFont(new Font("Times New Roman", Font.BOLD, 40));
            jLabel1.setHorizontalAlignment(SwingConstants.CENTER);
            jLabel1.setHorizontalTextPosition(SwingConstants.CENTER);
            // jLabel1.setText("Welcome to Health Care System");
            jLabel1.setIcon(new ImageIcon("healthcare_banner2.jpg"));
            //
            // jLabel2
            //
            jLabel2.setFont(new Font("Times New Roman", Font.BOLD, 18));
            jLabel2.setHorizontalAlignment(SwingConstants.CENTER);
            jLabel2.setHorizontalTextPosition(SwingConstants.CENTER);
            jLabel2.setText("Enter a web page URL to analyze:");
            //
            // jLabel3
            //
            jLabel3.setFont(new Font("Times New Roman", Font.BOLD, 18));
            jLabel3.setHorizontalAlignment(SwingConstants.CENTER);
            jLabel3.setHorizontalTextPosition(SwingConstants.CENTER);
            jLabel3.setText("Analyzed text from the above URL:");
            //
            // jLabel4
            //
            jLabel4.setFont(new Font("Times New Roman", Font.BOLD, 18));
            jLabel4.setText("Show");
            //
            // jLabel5
            //
```

```
jLabel5.setFont(new Font("Times New Roman", Font.BOLD, 18));
jLabel5.setText("in the analyzed text below.");
//
// jCheckBox1
//
jCheckBox1.setText("Treatments");
jCheckBox1.addItemListener(new ItemListener() {
	public void itemStateChanged(ItemEvent e) {
		jCheckBox1_itemStateChanged(e);
	}

});
//
// jCheckBox2
//
jCheckBox2.setText("Symptoms");
jCheckBox2.addItemListener(new ItemListener() {
	public void itemStateChanged(ItemEvent e) {
		jCheckBox2_itemStateChanged(e);
	}

});
//
// jScrollPane1
//
jScrollPane1.setViewportView(jTextArea1);
//
// jButton1
//
jButton1.setText("Submit");
jButton1.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {

		try {
			jButton1_actionPerformed(e);
		} catch (Exception e1) {
			e1.printStackTrace();
		}

	}

});
//
// jButton2
//
jButton2.setText("Cancel");
jButton2.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		jButton2_actionPerformed(e);
	}

});
//
// contentPane
//
Toolkit tk = Toolkit.getDefaultToolkit();
```

```
		int xSize = ((int) tk.getScreenSize().getWidth());
		int ySize = ((int) tk.getScreenSize().getHeight());

		contentPane.setLayout(null);
		addComponent(contentPane, jLabel1, 100, 24, (xSize - 300), 120);

		addComponent(contentPane, jLabel2, 50, 165, 300, 30);
		addComponent(contentPane, jTextField1, 350, 165, (xSize - 450), 30);

		addComponent(contentPane, jLabel4, 75, 220, 50, 30);
		addComponent(contentPane, jCheckBox1, 150, 220, 100, 30);
		addComponent(contentPane, jCheckBox2, 300, 220, 100, 30);
		addComponent(contentPane, jLabel5, 450, 220, 270, 30);

		addComponent(contentPane, jLabel3, 50, 270, 300, 30);
		addComponent(contentPane, jScrollPane1, 50, 300, (xSize - 150), 340);

		addComponent(contentPane, jButton1, (ySize / 2), 660, 83, 28);
		addComponent(contentPane, jButton2, (ySize / 2) + 200, 660, 83, 28);

		this.setTitle("HealthCare System");
		this.setLocation(new Point(0, 0));
		this.setSize(xSize - 20, ySize - 40);
	}

	/** Add Component Without a Layout Manager (Absolute Positioning) */
	private void addComponent(Container container, Component c, int x, int y,
			int width, int height) {
		c.setBounds(x, y, width, height);
		container.add(c);
	}

	private void jCheckBox1_itemStateChanged(ItemEvent e) {
		if ((e.getStateChange() == ItemEvent.SELECTED)) {
			// System.out.println("Treatments Selected");
			check1 = true;
		}
	}

	private void jCheckBox2_itemStateChanged(ItemEvent e) {
		if ((e.getStateChange() == ItemEvent.SELECTED)) {
			// System.out.println("Symptoms Selected");
			check2 = true;

		}

	}

	private void jButton1_actionPerformed(ActionEvent e) throws Exception {

		jButton1.setEnabled(false);

		// http://patientadvocare.blogspot.com/
		// http://e-patients.net/
		// http://www.protectpatientsblog.com/
		// http://advocacyforpatients.blogspot.com/
```

```
                // http://www.chroniclesofacancerpatient.com/
                // http://articles.cnn.com/2007-05-09/health/miles.blog_1_teen-cancer-rare-cancer-
blog?_s=PM:HEALTH
                // http://www.seattlepi.com/local/372201_cancerbloggers25.html
                // http://www2.mdanderson.org/cancerwise/2011/03/by-winston-huh-mdas-the.html
                // http://www.ritualsofhealing.com/blog/?Tag=cancer%20patients

                url = new URL(jTextField1.getText().toString().trim());

                final InputStream urlStream = url.openStream();
                final InputSource is = new InputSource(urlStream);

                final BoilerpipeSAXInput in = new BoilerpipeSAXInput(is);
                final TextDocument doc = in.getTextDocument();
                urlStream.close();

                HTMLEditorKit htmlKit = new HTMLEditorKit();
                jTextArea1.setContentType("text/html");
                jTextArea1.setEditorKit(htmlKit);
                jTextArea1.setEditable(false);

                if (check1 && !check2) {
                        // Drugs
                        String tempQ1 = "";
                        String q = CanolaExtractor.INSTANCE.getText(doc).toLowerCase()
                                        .replaceAll("[^a-zA-Z 0-9]+", "");

                        int off = 5000;
                        int qlength = q.length();

                        int xyz = 1;

                        while (off <= qlength) {
                                ArrayList<String> myArr = new ArrayList<String>();

                                myintArr = new ArrayList<Integer>();

                                String tempQ = q.substring(off - 5000, off);
                                String t = SearchIndex.searchCelebrityname("Drugs", tempQ);

                                t = t.replaceAll("[^A-Za-z0-9 ,##]", "");
                                String[] hashseparator = t.split("##,", -1);

                                for (int sep = 0; sep < hashseparator.length; sep++) {
                                        hashseparator[sep] = hashseparator[sep]
                                                        .replaceAll("[^A-Za-z0-9 ,.]", "")
                                                        .replaceAll("^\\s+", "").replaceAll("\\s+$", "");

                                        final String[] t1 = hashseparator[sep].split(",", -1);

                                        StringBuffer repl = new StringBuffer();
                                        repl.append("<A HREF=\""
                                                        + xyz
                                                        + "\" onMouseOver=\"\";><font size=\"4\"
face=\"verdana\" color=\"blue\">"
                                                        + t1[0] + "</font></A>");
```

```
                                        // webMD and PubMed links
                                        repl.append(" ( WebMD, ");
                                        repl.append("PubMed ) ");

                                        // WebMD
                                        StringBuffer link = new StringBuffer();
                                        StringBuffer link2 = new StringBuffer();

                                        StringBuffer web = new StringBuffer();
                                        StringBuffer pub = new StringBuffer();


        link.append("http://www.webmd.com/search/search_results/default.aspx?query=");

                                        link.append(((t1[0].replaceAll("Symptom: ", ""))
                                                        .replaceAll("^\\s+", "")).replaceAll("\\s+", "%20"));

                                        link.append("&sourceType=undefined");

                                        String we = "WebMD";
                                        web.append("<A HREF=\""
                                                        + link
                                                        + "\" onMouseOver=\"\";><font size=\"4\"
face=\"verdana\" color=\"green\">"
                                                        + we + "</font></A>");
                                        // PubMed

                                        link2.append("http://www.ncbi.nlm.nih.gov/pubmed?term=");

                                        link2.append(((t1[0].replaceAll("Symptom: ", ""))
                                                        .replaceAll("^\\s+", "")).replaceAll("\\s+", "%20"));

                                        String p = "PubMed";
                                        pub.append("<A HREF=\""
                                                        + link2
                                                        + "\" onMouseOver=\"\";><font size=\"4\"
face=\"verdana\" color=\"green\">"
                                                        + p + "</font></A>");

                                        hash3.put(xyz, t1[0]);

                                        myintArr.add(xyz);

                                        hash2.put(t1[0], myintArr);

                                        if (sep == 0) {
                                                String ty = repl.toString().replace("WebMD", web);
                                                ty = ty.replace("PubMed", pub);
                                                tempQ1 = tempQ1 + tempQ.replace(t1[0], ty);
                                        }

                                        final StringBuffer msg = new StringBuffer();
                                        if (t1[0].length() > 0) {
                                                msg.append("Drug Name: " + t1[0] + "\n\n Category: "
                                                                + t1[1] + "\n\n Number of Patients: " + t1[2]
```

```
                        + "\n\n Description: ");

                if (t1[3].length() > 100) {
                        int offset = 100;
                        while (offset <= t1[3].length()) {
                                msg.append(t1[3]
                                                .substring(offset - 100,
offset));
                                msg.append("\n");
                                offset += 100;
                        }
                        msg.append(t1[3].substring(offset - 100,
                                        t1[3].length()));
                } else if (t1[3].length() <= 100) {
                        msg.append(t1[3]);
                }

                if (t1[4].length() == 0) {
                        msg.append("\n\n Reasons taking " + t1[0] + ": "
                                        + "\n\n Percentage of Patients
taking "
                                        + t1[0] + " for reason " + t1[4] + ":
");
                        myArr.add(msg.toString() + "\n\n");
                } else {
                        t2check1 = (Integer
                                        .parseInt(SearchIndex.reason_hash

                .get(t1[0]).toString()));

                        t1check1 = ((Integer.parseInt(t1[5])) * 100)
                                        / t2check1;

                        msg.append("\n\n Reasons taking " + t1[0] + ": "
                                        + t1[4]
                                        + "\n\n Percentage of Patients
taking "
                                        + t1[0] + " for reason " + t1[4] + ":
"
                                        + t1check1);
                        myArr.add(msg.toString() + "\n\n");

                }
                if (t1[6].length() == 0) {
                        msg.append("\n\n Side Effect of taking " + t1[0]
                                        + ": "
                                        + "\n\n Percentage of Patients
suffering "
                                        + t1[6] + " side effect : ");
                        myArr.add(msg.toString() + "\n\n");
                } else {
                        t2check1 = (Integer.parseInt(SearchIndex.side_hash
                                        .get(t1[0]).toString()));

                        t1check1 = ((Integer.parseInt(t1[7])) * 100)
                                        / t2check1;
```

```
                                        msg.append("\n\n Side Effect of taking " + t1[0]
                                                + ": " + t1[6]
                                                + "\n\n Percentage of Patients
suffering "
                                                + t1[6] + " side effect : " +
t1check1);
                                        myArr.add(msg.toString() + "\n\n");

                                    }
                                    if (t1[8].length() == 0) {
                                        msg.append("\n\n Side Effect Severity: \n\n
Percentage of Patients reporting "
                                                + t1[8] + " severity: ");
                                        myArr.add(msg.toString() + "\n\n");
                                    } else {
                                        t2check1 = (Integer

            .parseInt(SearchIndex.sideproblem_hash.get(

            t1[0]).toString()));

                                        t1check1 = ((Integer.parseInt(t1[9])) * 100)
                                                / t2check1;

                                        msg.append("\n\n Side Effect Severity: " + t1[8]
                                                + "\n\n Percentage of Patients
reporting "
                                                + t1[8] + " severity: " + t1check1);
                                        myArr.add(msg.toString() + "\n\n");

                                    }
                                    if (t1[10].length() == 0) {
                                        msg.append("\n\n Reason for stopping consumption
of "
                                                + t1[0]
                                                + ": "
                                                + "\n\n Percentage of Patients who
stopped "
                                                + t1[0] + " drug: ");
                                        myArr.add(msg.toString() + "\n\n");
                                    } else {
                                        t2check1 = (Integer.parseInt(SearchIndex.stop_hash
                                                .get(t1[0]).toString()));

                                        t1check1 = ((Integer.parseInt(t1[11])) * 100)
                                                / t2check1;

                                        msg.append("\n\n Reason for stopping consumption
of "
                                                + t1[0]
                                                + ": "
                                                + t1[10]
                                                + "\n\n Percentage of Patients who
stopped "
                                                + t1[0] + " drug: " + t1check1);
```

```
                                        myArr.add(msg.toString() + "\n\n");

                                    }
                                    hash.put(xyz, msg + "\n\n");

                                    xyz += 1;
                                }

                            }

                            off += 5000;

                        }
                        String tempQ = q.substring(off - 5000, qlength);
                        String t = SearchIndex.searchCelebrityname("Drugs", tempQ);

                        ArrayList<String> myArr = new ArrayList<String>();

                        t = t.replaceAll("[^A-Za-z0-9 ,##]", "");
                        String[] hashseparator = t.split("##,", -1);

                        for (int sep = 0; sep < hashseparator.length; sep++) {
                            hashseparator[sep] = hashseparator[sep]
                                    .replaceAll("[^A-Za-z0-9 ,.]", "")
                                    .replaceAll("^\\s+", "").replaceAll("\\s+$", "");

                            final String[] t1 = hashseparator[sep].split(",", -1);
                            StringBuffer repl = new StringBuffer();
                            repl.append("<A HREF=\""
                                    + xyz
                                    + "\" onMouseOver=\"\";><font size=\"4\" face=\"verdana\"
color=\"blue\">"
                                    + t1[0] + "</font></A>");

                            // webMD and PubMed links
                            repl.append(" ( WebMD, ");
                            repl.append("PubMed ) ");

                            // WebMD
                            StringBuffer link = new StringBuffer();
                            StringBuffer link2 = new StringBuffer();

                            StringBuffer web = new StringBuffer();
                            StringBuffer pub = new StringBuffer();


        link.append("http://www.webmd.com/search/search_results/default.aspx?query=");

                            link.append(((t1[0].replaceAll("Symptom: ", "")).replaceAll(
                                    "^\\s+", "")).replaceAll("\\s+", "%20"));

                            link.append("&sourceType=undefined");

                            String we = "WebMD";
                            web.append("<A HREF=\""
                                    + link
```

```
                                        + "\" onMouseOver=\"\";><font size=\"4\" face=\"verdana\"
color=\"green\">"
                                        + we + "</font></A>");
                        // PubMed

                        link2.append("http://www.ncbi.nlm.nih.gov/pubmed?term=");

                        link2.append(((t1[0].replaceAll("Symptom: ", "")).replaceAll(
                                        "^\\s+", "")).replaceAll("\\s+", "%20"));

                        String p = "PubMed";
                        pub.append("<A HREF=\""
                                        + link2
                                        + "\" onMouseOver=\"\";><font size=\"4\" face=\"verdana\"
color=\"green\">"
                                        + p + "</font></A>");

                        hash3.put(xyz, t1[0]);

                        myintArr.add(xyz);

                        hash2.put(t1[0], myintArr);

                        if (sep == 0) {
                                String ty = repl.toString().replace("WebMD", web);
                                ty = ty.replace("PubMed", pub);
                                tempQ1 = tempQ1 + tempQ.replace(t1[0], ty);
                        }

                        final StringBuffer msg = new StringBuffer();
                        if (t1[0].length() > 0) {
                                msg.append("Drug Name: " + t1[0] + "\n\n Category: "
                                                + t1[1] + "\n\n Number of Patients: " + t1[2]
                                                + "\n\n Description: ");

                                if (t1[3].length() > 100) {
                                        int offset = 100;
                                        while (offset <= t1[3].length()) {
                                                msg.append(t1[3].substring(offset - 100, offset));
                                                msg.append("\n");
                                                offset += 100;
                                        }
                                        msg.append(t1[3].substring(offset - 100, t1[3].length()));
                                } else if (t1[3].length() <= 100) {
                                        msg.append(t1[3]);
                                }

                                if (t1[4].length() == 0) {
                                        msg.append("\n\n Reasons taking " + t1[0] + ": "
                                                        + "\n\n Percentage of Patients taking " +
t1[0]
                                                        + " for reason " + t1[4] + ": ");
                                        myArr.add(msg.toString() + "\n\n");
                                } else {

                                        t2check1 = (Integer.parseInt(SearchIndex.reason_hash
```

```
                                        .get(t1[0]).toString()));

                        t1check1 = ((Integer.parseInt(t1[5])) * 100) / t2check1;

                        msg.append("\n\n Reasons taking " + t1[0] + ": "
                                        + t1[4] + "\n\n Percentage of Patients taking
"
                                        + t1[0] + " for reason " + t1[4] + ": "
                                        + t1check1);
                        myArr.add(msg.toString() + "\n\n");

                }
                if (t1[6].length() == 0) {
                        msg.append("\n\n Side Effect of taking " + t1[0] + ": "
                                        + "\n\n Percentage of Patients suffering "
                                        + t1[6] + " side effect : ");
                        myArr.add(msg.toString() + "\n\n");
                } else {
                        t2check1 = (Integer.parseInt(SearchIndex.side_hash.get(
                                        t1[0]).toString()));

                        t1check1 = ((Integer.parseInt(t1[7])) * 100) / t2check1;

                        msg.append("\n\n Side Effect of taking " + t1[0] + ": "
                                        + t1[6]
                                        + "\n\n Percentage of Patients suffering "
                                        + t1[6] + " side effect : " + t1check1);
                        myArr.add(msg.toString() + "\n\n");

                }
                if (t1[8].length() == 0) {
                        msg.append("\n\n Side Effect Severity: \n\n Percentage of
Patients reporting "
                                        + t1[8] + " severity: ");
                        myArr.add(msg.toString() + "\n\n");
                } else {

                        t2check1 = (Integer

        .parseInt(SearchIndex.sideproblem_hash.get(
                                                        t1[0]).toString()));

                        t1check1 = ((Integer.parseInt(t1[9])) * 100) / t2check1;

                        msg.append("\n\n Side Effect Severity: " + t1[8]
                                        + "\n\n Percentage of Patients reporting "
                                        + t1[8] + " severity: " + t1check1);
                        myArr.add(msg.toString() + "\n\n");

                }
                if (t1[10].length() == 0) {
                        msg.append("\n\n Reason for stopping consumption of "
                                        + t1[0] + ": "
                                        + "\n\n Percentage of Patients who stopped "
                                        + t1[0] + " drug: ");
                        myArr.add(msg.toString() + "\n\n");
```

```
                                        } else {
                                                t2check1 = (Integer.parseInt(SearchIndex.stop_hash.get(
                                                        t1[0]).toString()));

                                                t1check1 = ((Integer.parseInt(t1[11])) * 100)
                                                        / t2check1;

                                                msg.append("\n\n Reason for stopping consumption of "
                                                        + t1[0] + ": " + t1[10]
                                                        + "\n\n Percentage of Patients who stopped "
                                                        + t1[0] + " drug: " + t1check1);
                                                myArr.add(msg.toString() + "\n\n");

                                        }
                                        hash.put(xyz, msg + "\n\n");

                                        xyz += 1;
                                }

                        }
                        jTextArea1.addHyperlinkListener(new HyperlinkListener() {
                                public void hyperlinkUpdate(HyperlinkEvent e) {
                                        if (e.getEventType() == HyperlinkEvent.EventType.ACTIVATED) {
                                                StringBuffer st = new StringBuffer();

                                                if (e.getDescription().startsWith("h")) {
                                                        try {
                                                                java.awt.Desktop.getDesktop()

        .browse(java.net.URI.create(e

        .getDescription()));

                                                        } catch (IOException e1) {
                                                                e1.printStackTrace();
                                                        }

                                                } else

                                                {
                                                        String hshtemp = hash3.get(

        (Integer.parseInt(e.getDescription())))
                                                                        .toString();
                                                        ArrayList<Integer> temp = (ArrayList<Integer>)
hash2
                                                                        .get(hshtemp);

                                                        Integer w;
                                                        StringBuilder new1 = new StringBuilder();

                                                        int time = 0;

                                                        for (int h = 0; h < temp.size(); h++) {
                                                                w = temp.get(h);
```

```
                String[] er = hash
                                .get(Integer.parseInt(w +
""))
                                .toString().split("\n");

                String temp_er = er[0].replace("Drug Name:
",
                                "");

                if (hshtemp.equals(temp_er)) {

                        new1.append(hash.get(w).toString());
                                time += 1;

                }
        }

        String[] shw = new1.toString().split("\n\n");

        if (shw[0].length() > 0) {
                st.append(shw[0].replaceAll("^\\s+", "") +
"\n");
        }

        if (shw[1].length() > 0) {
                st.append(shw[1].replaceAll("^\\s+", "") +
"\n");
        }
        if (shw[2].length() > 0) {
                st.append(shw[2].replaceAll("^\\s+", "")
                                .replaceFirst("\n", "") +
"\n");
        }
        if (shw[3].length() > 0) {
                st.append(shw[3].replaceAll("^\\s+", "") +
"\n");
        }

        for (int y = 0; y < time; y++) {

                if (y == 0) {

                        columntext45.removeAll(columntext45);

                        columntext67.removeAll(columntext67);

                        columntext89.removeAll(columntext89);

                        columntext1011.removeAll(columntext1011);

                                columncount = 0;
                }

                String[] shw_symp = new1.toString().split(
                                "\n\n");
```

```
+ (y * 12)]



shw_symp[5 + (y * 12)]

");


        columntext45.add(shw_temp4[1]

        .replaceAll("^\\s+", "")

shw_temp5[1].replaceAll(

        "^\\s+", ""));




+ (y * 12)]



shw_symp[7 + (y * 12)]

");


        columntext67.add(shw_temp6[1]

        .replaceAll("^\\s+", "")

shw_temp7[1].replaceAll(

        "^\\s+", ""));




+ (y * 12)]
```

```
if (shw_symp[4 + (y * 12)].length() > 0) {
        String[] shw_temp4 = shw_symp[4

                        .split(": ");
        if (shw_temp4.length > 1) {

                String[] shw_temp5 =

                                .split(":

                columncount += 1;



                                + "\t"
                                +


        }
}

if (shw_symp[6 + (y * 12)].length() > 0) {
        String[] shw_temp6 = shw_symp[6

                        .split(": ");
        if (shw_temp6.length > 1) {

                String[] shw_temp7 =

                                .split(":

                columncount += 1;



                                + "\t"
                                +


        }

}

if (shw_symp[8 + (y * 12)].length() > 0) {
        String[] shw_temp8 = shw_symp[8

                        .split(": ");
```

```
shw_symp[9 + (y * 12)]
");


        columntext89.add(shw_temp8[1]
        .replaceAll("^\\s+", "")
shw_temp9[1].replaceAll(
        "^\\s+", ""));




shw_symp[10 + (y * 12)]


shw_symp[11 + (y * 12)]
");


        columntext1011.add(shw_temp10[1]
        .replaceAll("^\\s+", "")
shw_temp11[1].replaceAll(
        "^\\s+", ""));


                if (shw_temp8.length > 1) {
                        String[] shw_temp9 =
                                        .split(":
                        columncount += 1;


                                                        + "\t"
                                                        +


                }
        }
        if (shw_symp[10 + (y * 12)].length() > 0) {
                String[] shw_temp10 =
                                .split(": ");
                if (shw_temp10.length > 1) {
                        String[] shw_temp11 =
                                        .split(":
                        columncount += 1;


                                                        + "\t"
                                                        +


                }
        }
}
JTextArea textArea = new JTextArea();

textArea.setEditable(false);
textArea.setText(st.toString().replaceAll("^\\s+",
                "")
                + "\n\n\t");
```

```
                                                JTextPane textPane = new JTextPane();
                                                textPane.setSize(700, 300);
                                                textPane.setEditable(false);
                                                textPane.setPreferredSize(textPane.getSize());

                                                textPane.scrollRectToVisible(new Rectangle(0,
                                                                textPane.getWidth() - 2, 1, 1));

                                                StyledDocument doc =
textPane.getStyledDocument();
                                                createStyles(doc);
                                                setContent(doc, st.toString());
                                                styleContent(doc);

                                                JScrollPane scrollPane = new JScrollPane(textPane);
                                                scrollPane.setSize(15, 80);

                                                JOptionPane.showMessageDialog(null, scrollPane,
                                                                "Details of Matched Keyword",

        JOptionPane.INFORMATION_MESSAGE);

                                                        }
                                                }
                                        }

                                        private void createStyles(StyledDocument doc) {
                                                Style baseStyle = doc.addStyle("base", null);
                                                StyleConstants.setFontFamily(baseStyle,
                                                                "Lucida Sans Unicode");
                                                StyleConstants.setFontSize(baseStyle, 12);
                                                StyleConstants.setFirstLineIndent(baseStyle, 20f);
                                                StyleConstants.setLeftIndent(baseStyle, 10f);

                                                Style style = doc.addStyle("table1", null);
                                                StyleConstants.setComponent(style, getTableComponent1());

                                                style = doc.addStyle("table2", null);
                                                StyleConstants.setComponent(style, getTableComponent2());

                                                style = doc.addStyle("table3", null);
                                                StyleConstants.setComponent(style, getTableComponent3());

                                                style = doc.addStyle("table4", null);
                                                StyleConstants.setComponent(style, getTableComponent4());

                                                style = doc.addStyle("tableParagraph", null);
                                                StyleConstants.setLeftIndent(style, 35f);
                                                StyleConstants.setRightIndent(style, 35f);
                                                StyleConstants.setSpaceAbove(style, 15f);
                                                StyleConstants.setSpaceBelow(style, 15f);
                                        }

                                        private JScrollPane getTableComponent1() {
                                                JTable table = new JTable(getModel1());
```

```
        TableColumn col = table.getColumnModel().getColumn(0);
        col.setHeaderRenderer(new MyTableHeaderRenderer());

        col = table.getColumnModel().getColumn(1);
        col.setHeaderRenderer(new MyTableHeaderRenderer1());
        table.setEnabled(false);

        Dimension d = table.getPreferredSize();
        d.width = 300;

        table.setPreferredScrollableViewportSize(d);
        return new JScrollPane(table);
    }

    private JScrollPane getTableComponent2() {
        JTable table = new JTable(getModel2());

        TableColumn col = table.getColumnModel().getColumn(0);
        col.setHeaderRenderer(new MyTableHeaderRenderer2());

        col = table.getColumnModel().getColumn(1);
        col.setHeaderRenderer(new MyTableHeaderRenderer3());
        table.setEnabled(false);

        Dimension d = table.getPreferredSize();
        d.width = 300;

        table.setPreferredScrollableViewportSize(d);
        return new JScrollPane(table);
    }

    private JScrollPane getTableComponent3() {
        JTable table = new JTable(getModel3());

        TableColumn col = table.getColumnModel().getColumn(0);
        col.setHeaderRenderer(new MyTableHeaderRenderer4());

        col = table.getColumnModel().getColumn(1);
        col.setHeaderRenderer(new MyTableHeaderRenderer5());
        table.setEnabled(false);

        Dimension d = table.getPreferredSize();
        d.width = 300;

        table.setPreferredScrollableViewportSize(d);
        return new JScrollPane(table);
    }

    private JScrollPane getTableComponent4() {
        JTable table = new JTable(getModel4());

        TableColumn col = table.getColumnModel().getColumn(0);
        col.setHeaderRenderer(new MyTableHeaderRenderer6());

        col = table.getColumnModel().getColumn(1);
```

```
                col.setHeaderRenderer(new MyTableHeaderRenderer7());
                table.setEnabled(false);

                Dimension d = table.getPreferredSize();
                d.width = 300;

                table.setPreferredScrollableViewportSize(d);
                return new JScrollPane(table);
        }

        public AbstractTableModel getModel1() {
                int t = 0;

                // System.out.println("In 45");
                for (int ty = 0; ty < columntext45.size(); ty++) {
                        String[] rowsplit = columntext45.get(ty)
                                        .split("\t", -1);

                        for (int tz = 0; tz < rowsplit.length; tz++) {
                                rowceil.add(rowsplit[tz]);

                                rowval.put(String.valueOf(ty) + (tz), rowsplit[tz]);
                        }

                }

                final HashMap<String, Double> map = new HashMap<String,
Double>();
                ValueComparator bvc = new ValueComparator(map);
                final TreeMap<String, Double> sorted_map = new TreeMap(bvc);

                for (int y = 0; y < columntext45.size(); y++) {
                        map.put(rowval.get(String.valueOf(y) + (0)).toString(),
                                        Double.parseDouble(rowval.get(
                                                        String.valueOf(y) +
(1)).toString()));
                }

                for (String key : map.keySet()) {
                }

                sorted_map.putAll(map);

                int y1 = 0;
                for (String key : sorted_map.keySet()) {

                        if (sorted_map.get(key) != null) {
                                float p = (float) Math.pow(10, 2);
                                double Rval = sorted_map.get(key) * p;
                                float tmp = Math.round(Rval);
                                String typ = (float) tmp / p + "";

                                ceilval.put(String.valueOf(y1) + (0), key);
                                ceilval.put(String.valueOf(y1) + (1), typ);
                        } else {
                                ceilval.put(String.valueOf(y1) + (0), key);
```

```
                                        ceilval.put(String.valueOf(y1) + (1), "N/A");
                                }
                                y1 += 1;
                        }

                        if (columntext45.size() <= 5) {
                                return new AbstractTableModel() {

                                        public int getColumnCount() {
                                                return 2;
                                        }

                                        public int getRowCount() {
                                                return columntext45.size();
                                        }

                                        public Object getValueAt(int row, int col) {

                                                return ceilval.get(String.valueOf(row) +
(col));


                                        }

                                };
                        } else {
                                return new AbstractTableModel() {

                                        public int getColumnCount() {
                                                return 2;
                                        }

                                        public int getRowCount() {
                                                return 5;
                                        }

                                        public Object getValueAt(int row, int col) {
                                                return ceilval.get(String.valueOf(row) +
(col));


                                        }

                                };
                        }

                }

                public AbstractTableModel getModel2() {
                        int t = 0;

                        for (int ty = 0; ty < columntext67.size(); ty++) {
                                String[] rowsplit = columntext67.get(ty)
                                                .split("\t", -1);

                                for (int tz = 0; tz < rowsplit.length; tz++) {
                                        rowceil1.add(rowsplit[tz]);
```

```
                                        rowval1.put(String.valueOf(ty) + (tz), rowsplit[tz]);
                                }

                        }

                        final HashMap<String, Double> map = new HashMap<String,
Double>();
                        ValueComparator bvc = new ValueComparator(map);
                        final TreeMap<String, Double> sorted_map = new TreeMap(bvc);

                        for (int y = 0; y < columntext67.size(); y++) {
                                map.put(rowval1.get(String.valueOf(y) + (0)).toString(),
                                                Double.parseDouble(rowval1.get(
                                                                String.valueOf(y) +
(1)).toString()));
                        }

                        for (String key : map.keySet()) {
                        }

                        sorted_map.putAll(map);

                        int y1 = 0;
                        for (String key : sorted_map.keySet()) {
                                if (sorted_map.get(key) != null) {
                                        float p = (float) Math.pow(10, 2);
                                        double Rval = sorted_map.get(key) * p;
                                        float tmp = Math.round(Rval);

                                        String typ = (float) tmp / p + "";

                                        ceilval1.put(String.valueOf(y1) + (0), key);
                                        ceilval1.put(String.valueOf(y1) + (1), typ);
                                } else {
                                        ceilval1.put(String.valueOf(y1) + (0), key);
                                        ceilval1.put(String.valueOf(y1) + (1), "N/A");
                                }
                                y1 += 1;
                        }
                        if (columntext67.size() <= 5) {
                                return new AbstractTableModel() {

                                        public int getColumnCount() {
                                                return 2;
                                        }

                                        public int getRowCount() {
                                                return columntext67.size();
                                        }

                                        public Object getValueAt(int row, int col) {
                                                return ceilval1.get(String.valueOf(row) +
(col));


                                        }
```

```
                };
            } else {
                return new AbstractTableModel() {

                    public int getColumnCount() {
                        return 2;
                    }

                    public int getRowCount() {
                        return 5;
                    }

                    public Object getValueAt(int row, int col) {
                        return ceilval1.get(String.valueOf(row) +
(col));


                    }

                };
            }

        }

        public AbstractTableModel getModel3() {
            int t = 0;

            // System.out.println("In 89");
            for (int ty = 0; ty < columntext89.size(); ty++) {
                String[] rowsplit = columntext89.get(ty)
                        .split("\t", -1);
                for (int tz = 0; tz < rowsplit.length; tz++) {
                    rowceil2.add(rowsplit[tz]);

                    rowval2.put(String.valueOf(ty) + (tz), rowsplit[tz]);
                }

            }

            final HashMap<String, Double> map = new HashMap<String,
Double>();
            ValueComparator bvc = new ValueComparator(map);
            final TreeMap<String, Double> sorted_map = new TreeMap(bvc);

            for (int y = 0; y < columntext89.size(); y++) {
                map.put(rowval2.get(String.valueOf(y) + (0)).toString(),
                        Double.parseDouble(rowval2.get(
                                String.valueOf(y) +
(1)).toString()));
            }
            for (String key : map.keySet()) {
            }

            sorted_map.putAll(map);
            int y1 = 0;
            for (String key : sorted_map.keySet()) {
                if (sorted_map.get(key) != null) {
```

```
                                        float p = (float) Math.pow(10, 2);
                                        double Rval = sorted_map.get(key) * p;
                                        float tmp = Math.round(Rval);
                                        String typ = (float) tmp / p + "";

                                        ceilval2.put(String.valueOf(y1) + (0), key);
                                        ceilval2.put(String.valueOf(y1) + (1), typ);
                                } else {
                                        ceilval2.put(String.valueOf(y1) + (0), key);
                                        ceilval2.put(String.valueOf(y1) + (1), "N/A");
                                }
                                y1 += 1;
                        }

                        if (columntext89.size() <= 5) {
                                return new AbstractTableModel() {

                                        public int getColumnCount() {
                                                return 2;
                                        }

                                        public int getRowCount() {
                                                return columntext89.size();
                                        }

                                        public Object getValueAt(int row, int col) {
                                                return ceilval2.get(String.valueOf(row) +
(col));


                                        }

                                };
                        } else {
                                return new AbstractTableModel() {

                                        public int getColumnCount() {
                                                return 2;
                                        }

                                        public int getRowCount() {
                                                return 5;
                                        }

                                        public Object getValueAt(int row, int col) {
                                                return ceilval2.get(String.valueOf(row) +
(col));


                                        }

                                };
                        }

                }

                public AbstractTableModel getModel4() {
                        int t = 0;
```

```
for (int ty = 0; ty < columntext1011.size(); ty++) {
	String[] rowsplit = columntext1011.get(ty).split("\t",
			-1);
	for (int tz = 0; tz < rowsplit.length; tz++) {
		rowceil3.add(rowsplit[tz]);

		rowval3.put(String.valueOf(ty) + (tz), rowsplit[tz]);
	}

}

final HashMap<String, Double> map = new HashMap<String,
Double>();
ValueComparator bvc = new ValueComparator(map);
final TreeMap<String, Double> sorted_map = new TreeMap(bvc);

for (int y = 0; y < columntext1011.size(); y++) {
	map.put(rowval3.get(String.valueOf(y) + (0)).toString(),
			Double.parseDouble(rowval3.get(
					String.valueOf(y) +
(1)).toString()));
}
for (String key : map.keySet()) {
}

sorted_map.putAll(map);
int y1 = 0;
for (String key : sorted_map.keySet()) {
	if (sorted_map.get(key) != null) {
		float p = (float) Math.pow(10, 2);
		double Rval = sorted_map.get(key) * p;
		float tmp = Math.round(Rval);
		String typ = (float) tmp / p + "";

		ceilval3.put(String.valueOf(y1) + (0), key);
		ceilval3.put(String.valueOf(y1) + (1), typ);
	} else {
		ceilval3.put(String.valueOf(y1) + (0), key);
		ceilval3.put(String.valueOf(y1) + (1), "N/A");
	}
	y1 += 1;
}

if (columntext1011.size() <= 5) {
	return new AbstractTableModel() {

		public int getColumnCount() {
			return 2;
		}

		public int getRowCount() {
			return columntext1011.size();
		}

		public Object getValueAt(int row, int col) {
```

```
                                        return ceilval3.get(String.valueOf(row) +
(col));
                                    }

                                };
                        } else {
                                return new AbstractTableModel() {

                                    public int getColumnCount() {
                                        return 2;
                                    }

                                    public int getRowCount() {
                                        return 5;
                                    }

                                    public Object getValueAt(int row, int col) {
                                        return ceilval3.get(String.valueOf(row) +
(col));


                                    }

                                };
                        }

                }

                class ValueComparator implements Comparator {

                        Map base;

                        public ValueComparator(Map base) {
                                this.base = base;
                        }

                        public int compare(Object a, Object b) {

                                if ((Double) base.get(a) < (Double) base.get(b)) {
                                        return 1;
                                } else if ((Double) base.get(a) == (Double) base.get(b)) {
                                        return 0;
                                } else {
                                        return -1;
                                }
                        }
                }

                private void setContent(StyledDocument doc, String text) {
                        try {
                                doc.insertString(0, text, doc.getStyle("base"));
                                doc.insertString(doc.getLength(), "\n", null);

                                doc.insertString(doc.getLength(), "\n",
                                                doc.getStyle("table1"));
                                doc.insertString(doc.getLength(), "\n", null);
                                doc.insertString(doc.getLength(), "\n", null);
```

```
                                        doc.insertString(doc.getLength(), "\n",
                                                        doc.getStyle("table2"));
                                        doc.insertString(doc.getLength(), "\n", null);
                                        doc.insertString(doc.getLength(), "\n", null);

                                        doc.insertString(doc.getLength(), "\n",
                                                        doc.getStyle("table3"));
                                        doc.insertString(doc.getLength(), "\n", null);
                                        doc.insertString(doc.getLength(), "\n", null);

                                        doc.insertString(doc.getLength(), "\n",
                                                        doc.getStyle("table4"));
                                        doc.insertString(doc.getLength(), "\n", null);
                                        doc.insertString(doc.getLength(), "\n", null);

                                } catch (BadLocationException e) {
                                }
                        }

                        private void styleContent(StyledDocument doc) {
                                Style style = doc.getStyle("base");
                                doc.setLogicalStyle(0, style);
                        }
                });
                jTextArea1.setText(tempQ1);
        } else if (check2 && !check1) {
                int r = 0;
                String tempQ1 = "";
                String q = CanolaExtractor.INSTANCE.getText(doc).toLowerCase()
                                .replaceAll("[^a-zA-Z 0-9]+", "");
                int off = 5000;
                int qlength = q.length();

                int xyz = 1;

                while (off <= qlength) {
                        ArrayList<String> myArr = new ArrayList<String>();

                        myintArr = new ArrayList<Integer>();

                        String tempQ = q.substring(off - 5000, off);
                        String t = SearchIndex.searchCelebrityname("Symptoms", tempQ);

                        t = t.replaceAll("[^A-Za-z0-9 ,##]", "");
                        String[] hashseparator = t.split("##,", -1);

                        for (int sep = 0; sep < hashseparator.length; sep++) {
                                hashseparator[sep] = hashseparator[sep]
                                                .replaceAll("[^A-Za-z0-9 ,.]", "")
                                                .replaceAll("^\\s+", "").replaceAll("\\s+$", "");

                                final String[] t1 = hashseparator[sep].split(",", -1);

                                StringBuffer repl = new StringBuffer();
                                repl.append("<A HREF=\""
```

```
                                                + xyz
                                                + "\" onMouseOver=\"\";><font size=\"4\"
face=\"verdana\" color=\"blue\">"
                                                + t1[0] + "</font></A>");

                                        // webMD and PubMed links
                                        repl.append(" ( WebMD, ");
                                        repl.append("PubMed ) ");

                                        // WebMD
                                        StringBuffer link = new StringBuffer();
                                        StringBuffer link2 = new StringBuffer();

                                        StringBuffer web = new StringBuffer();
                                        StringBuffer pub = new StringBuffer();


        link.append("http://www.webmd.com/search/search_results/default.aspx?query=");

                                        link.append(((t1[0].replaceAll("Symptom: ", ""))
                                                        .replaceAll("^\\s+", "")).replaceAll("\\s+", "%20"));

                                        link.append("&sourceType=undefined");

                                        String we = "WebMD";
                                        web.append("<A HREF=\""
                                                        + link
                                                        + "\" onMouseOver=\"\";><font size=\"4\"
face=\"verdana\" color=\"green\">"
                                                        + we + "</font></A>");
                                        // PubMed

                                        link2.append("http://www.ncbi.nlm.nih.gov/pubmed?term=");

                                        link2.append(((t1[0].replaceAll("Symptom: ", ""))
                                                        .replaceAll("^\\s+", "")).replaceAll("\\s+", "%20"));

                                        String p = "PubMed";
                                        pub.append("<A HREF=\""
                                                        + link2
                                                        + "\" onMouseOver=\"\";><font size=\"4\"
face=\"verdana\" color=\"green\">"
                                                        + p + "</font></A>");

                                        hash3.put(xyz, t1[0]);

                                        myintArr.add(xyz);
                                        hash2.put(t1[0], myintArr);

                                        if (sep == 0) {
                                                String ty = repl.toString().replace("WebMD", web);
                                                ty = ty.replace("PubMed", pub);
                                                tempQ1 = tempQ1 + tempQ.replace(t1[0], ty);

                                        }
```

```
                        final StringBuffer msg = new StringBuffer();
                        if (t1[0].length() > 0) {
                                msg.append("Symptom: " + t1[0]
                                                + "\n\n Number of Patients: " + t1[1]
                                                + "\n\n Description: ");

                                if (t1[2].length() > 100) {
                                        int offset = 100;
                                        while (offset <= t1[2].length()) {
                                                msg.append(t1[2]
                                                                .substring(offset - 100,
offset));

                                                msg.append("\n");
                                                offset += 100;
                                        }
                                        msg.append(t1[2].substring(offset - 100,
                                                        t1[2].length()));
                                } else if (t1[2].length() <= 100) {
                                        msg.append(t1[2]);
                                }

                                if (t1[3].length() == 0) {
                                        msg.append("\n\n What Patients take for " + t1[0]
                                                        + ": "
                                                        + "\n\n Percentage of Patients
taking "
                                                        + t1[3] + " for treating " + t1[0] + ":
");
                                        myArr.add(msg.toString() + "\n\n");
                                } else {

                                        t2 = (Integer.parseInt(SearchIndex.hashMap.get(
                                                        t1[0]).toString()));

                                        t11 = ((Integer.parseInt(t1[3])) * 100) / t2;

                                        msg.append("\n\n What Patients take for " + t1[0]
                                                        + ": " + t1[4]
                                                        + "\n\n Percentage of Patients
taking "
                                                        + t1[3] + " for treating " + t1[0] + ":
"
                                                        + t11);

                                        myArr.add(msg.toString().replaceAll("^\\s+", "")
                                                        + "\n");
                                }
                                hash.put(xyz, msg + "\n\n");

                                xyz += 1;

                        }

                }

                off += 5000;
```

```
                        }

                        String tempQ = q.substring(off - 5000, qlength);
                        String t = SearchIndex.searchCelebrityname("Symptoms", tempQ);

                        ArrayList<String> myArr = new ArrayList<String>();

                        t = t.replaceAll("[^A-Za-z0-9 ,##]", "");
                        String[] hashseparator = t.split("##,", -1);

                        for (int sep = 0; sep < hashseparator.length; sep++) {
                                hashseparator[sep] = hashseparator[sep]
                                                .replaceAll("[^A-Za-z0-9 ,.]", "")
                                                .replaceAll("^\\s+", "").replaceAll("\\s+$", "");

                                final String[] t1 = hashseparator[sep].split(",", -1);

                                StringBuffer repl = new StringBuffer();
                                repl.append("<A HREF=\""
                                                + xyz
                                                + "\" onMouseOver=\"\";><font size=\"4\" face=\"verdana\"
color=\"blue\">"
                                                + t1[0] + "</font></A>");

                                // webMD and PubMed links
                                repl.append(" ( WebMD, ");
                                repl.append("PubMed ) ");

                                // WebMD
                                StringBuffer link = new StringBuffer();
                                StringBuffer link2 = new StringBuffer();

                                StringBuffer web = new StringBuffer();
                                StringBuffer pub = new StringBuffer();


        link.append("http://www.webmd.com/search/search_results/default.aspx?query=");

                                link.append(((t1[0].replaceAll("Symptom: ", "")).replaceAll(
                                                "^\\s+", "")).replaceAll("\\s+", "%20"));

                                link.append("&sourceType=undefined");

                                String we = "WebMD";
                                web.append("<A HREF=\""
                                                + link
                                                + "\" onMouseOver=\"\";><font size=\"4\" face=\"verdana\"
color=\"green\">"
                                                + we + "</font></A>");
                                // PubMed

                                link2.append("http://www.ncbi.nlm.nih.gov/pubmed?term=");

                                link2.append(((t1[0].replaceAll("Symptom: ", "")).replaceAll(
                                                "^\\s+", "")).replaceAll("\\s+", "%20"));
```

```
String p = "PubMed";
pub.append("<A HREF=\""
                + link2
                + "\" onMouseOver=\"\";><font size=\"4\" face=\"verdana\"
color=\"green\">"
                + p + "</font></A>");

hash3.put(xyz, t1[0]);

myintArr.add(xyz);

hash2.put(t1[0], myintArr);

if (sep == 0) {
        String ty = repl.toString().replace("WebMD", web);
        ty = ty.replace("PubMed", pub);
        tempQ1 = tempQ1 + tempQ.replace(t1[0], ty);

}
final StringBuffer msg = new StringBuffer();
if (t1[0].length() > 0) {
        msg.append("Symptom: " + t1[0]
                        + "\n\n Number of Patients: " + t1[1]
                        + "\n\n Description: ");

        if (t1[2].length() > 100) {
                int offset = 100;
                while (offset <= t1[2].length()) {
                        msg.append(t1[2].substring(offset - 100, offset));
                        msg.append("\n");
                        offset += 100;
                }
                msg.append(t1[2].substring(offset - 100, t1[2].length()));
        } else if (t1[2].length() <= 100) {
                msg.append(t1[2]);
        }

        if (t1[3].length() == 0) {
                msg.append("\n\n What Patients take for " + t1[0]
                                + ": " + "\n\n Percentage of Patients taking "
                                + t1[3] + " for treating " + t1[0] + ": ");
                myArr.add(msg.toString().replaceAll("^\\s+", "") + "\n");
        } else {

                t2 = (Integer.parseInt(SearchIndex.hashMap.get(t1[0])
                                .toString()));

                t11 = ((Integer.parseInt(t1[3])) * 100) / t2;

                msg.append("\n\n What Patients take for " + t1[0]
                                + ": " + t1[4]
                                + "\n\n Percentage of Patients taking " +
t1[3]
                                + " for treating " + t1[0] + ": " + t11);
```

```
                                        myArr.add(msg.toString().replaceAll("^\\s+", "") + "\n");
                                }

                                hash.put(xyz, msg + "\n\n");

                                xyz += 1;
                        }

                }

                jTextArea1.addHyperlinkListener(new HyperlinkListener() {
                        public void hyperlinkUpdate(HyperlinkEvent e) {
                                if (e.getEventType() == HyperlinkEvent.EventType.ACTIVATED) {

                                        StringBuffer st = new StringBuffer();

                                        if (e.getDescription().startsWith("h")) {
                                                try {
                                                        java.awt.Desktop.getDesktop()

        .browse(java.net.URI.create(e

        .getDescription()));

                                                } catch (IOException e1) {
                                                        e1.printStackTrace();
                                                }

                                        } else {

                                                String hshtemp = hash3.get(

        (Integer.parseInt(e.getDescription())))
                                                                .toString();

                                                ArrayList<Integer> temp = (ArrayList<Integer>)
hash2
                                                                .get(hshtemp);

                                                Integer w;
                                                StringBuilder new1 = new StringBuilder();

                                                StringBuffer link = new StringBuffer();
                                                StringBuffer link2 = new StringBuffer();

                                                StringBuffer web = new StringBuffer();
                                                StringBuffer pub = new StringBuffer();

                                                textArea = new JTextPane();

                                                // new
                                                HTMLEditorKit htmlKit = new HTMLEditorKit();
                                                textArea.setContentType("text/html");
                                                textArea.setStyledDocument(new
javax.swing.text.html.HTMLDocument());
                                                textArea.setEditorKit(htmlKit);
```

```
                                                textArea.setEditable(false);
                                                // end new

                                                int time = 0;

                                                for (int h = 0; h < temp.size() - 1; h++) {
                                                        w = temp.get(h);

                                                        String[] er = hash
                                                                        .get(Integer.parseInt(w +
""))
                                                                        .toString().split("\n");

                                                        String temp_er = er[0].replace("Symptom: ",
"");

                                                        if (hshtemp.equals(temp_er)) {

        new1.append(hash.get(w).toString());
                                                                time += 1;

                                                        }
                                                }

                                                String[] shw = new1.toString().split("\n\n");

                                                if (shw[0].length() > 0) {
                                                        st.append(shw[0].replaceAll("^\\s+", "") +
"\n");
                                                }

                                                if (shw[1].length() > 0) {
                                                        st.append(shw[1].replaceAll("^\\s+", "") +
"\n");
                                                }
                                                if (shw[2].length() > 0) {
                                                        st.append(shw[2].replaceAll("^\\s+", "")
                                                                        .replaceFirst("\n", "") +
"\n");
                                                }
                                                // webMD and PubMed links
                                                // WebMD
                                                st.append("WebMD: ");

        link.append("http://www.webmd.com/search/search_results/default.aspx?query=");

                                                link.append(((shw[0].replaceAll("Symptom: ", ""))
                                                                .replaceAll("^\\s+", "")).replaceAll(
                                                                "\\s+", "%20"));

                                                link.append("&sourceType=undefined");

                                                st.append(link + "\n");
                                                String we = "WebMd";
                                                web.append("<A HREF=\""
```

```
                                                + link
                                                + "\" onMouseOver=\"\";><font
size=\"4\" face=\"verdana\" color=\"blue\">"
                                                + we + "</font></A>");
                                        // PubMed
                                        st.append("PubMed: ");


        link2.append("http://www.ncbi.nlm.nih.gov/pubmed?term=");

                                        link2.append(((shw[0].replaceAll("Symptom: ", ""))
                                                .replaceAll("^\\s+", "")).replaceAll(
                                                "\\s+", "%20"));

                                        st.append(link2 + "\n");
                                        String p = "PubMed";
                                        pub.append("<A HREF=\""
                                                + link2
                                                + "\" onMouseOver=\"\";><font
size=\"4\" face=\"verdana\" color=\"blue\">"
                                                + p + "</font></A>");

                                        for (int y = 0; y < time; y++) {

                                                if (y == 0) {

        columntext.removeAll(columntext);
                                                        columncount = 0;
                                                }

                                                String[] shw_symp = new1.toString().split(
                                                                "\n\n");

                                                if (shw_symp[3 + (y * 5)].length() > 0) {

                                                        String[] shw_temp3 = shw_symp[3
+ (y * 5)]
                                                                        .split(": ");

                                                        String[] shw_temp4 = shw_symp[4
+ (y * 5)]
                                                                        .split(": ");

                                                        columncount += 1;


        columntext.add(shw_temp3[1].replaceAll(
                                                                        "^\\s+", "")
                                                                        + "\t"
                                                                        +
shw_temp4[1].replaceAll("^\\s+",
        ""));
                                                }
                                        }
```

```
                                                textArea.setEditable(false);

                                                String tr1 = st.toString().replace("WebMD", web);

                                                String tr2 = tr1.replace("PubMed", pub);

                                                textArea.setText(st.toString().replaceAll("^\\s+",
                                                                "")
                                                                + "\n\n\t");

                                                textPane = new JTextPane();
                                                textPane.setSize(700, 300);
                                                textPane.setEditable(false);
                                                textPane.setPreferredSize(textPane.getSize());

                                                textPane.scrollRectToVisible(new Rectangle(0,
                                                                textPane.getWidth() - 2, 1, 1));

                                                StyledDocument doc =
textPane.getStyledDocument();

                                                createStyles(doc);
                                                setContent(doc, st.toString());
                                                styleContent(doc);

                                                JScrollPane scrollPane = new JScrollPane(textPane);
                                                scrollPane.setSize(15, 80);

                                                JOptionPane.showMessageDialog(null, scrollPane,
                                                                "Details of Matched Keyword",

        JOptionPane.INFORMATION_MESSAGE);
                                                        }
                                                }
                                        }

                                        private void createStyles(StyledDocument doc) {
                                                Style baseStyle = doc.addStyle("base", null);
                                                StyleConstants.setFontFamily(baseStyle,
                                                                "Lucida Sans Unicode");
                                                StyleConstants.setFontSize(baseStyle, 12);
                                                StyleConstants.setFirstLineIndent(baseStyle, 20f);
                                                StyleConstants.setLeftIndent(baseStyle, 10f);

                                                Style style = doc.addStyle("bold", baseStyle);
                                                StyleConstants.setBold(style, true);

                                                style = doc.addStyle("italic", baseStyle);
                                                StyleConstants.setItalic(style, true);

                                                style = doc.addStyle("blue", baseStyle);
                                                StyleConstants.setForeground(style, Color.blue);

                                                style = doc.addStyle("underline", baseStyle);
                                                StyleConstants.setUnderline(style, true);

                                                style = doc.addStyle("green", baseStyle);
```

```
                StyleConstants.setForeground(style, Color.green.darker());
                StyleConstants.setUnderline(style, true);

                style = doc.addStyle("highlight", baseStyle);
                StyleConstants.setForeground(style, Color.yellow);
                StyleConstants.setBackground(style, Color.black);

                style = doc.addStyle("table", null);
                StyleConstants.setComponent(style, getTableComponent());

                style = doc.addStyle("tableParagraph", null);
                StyleConstants.setLeftIndent(style, 35f);
                StyleConstants.setRightIndent(style, 35f);
                StyleConstants.setSpaceAbove(style, 15f);
                StyleConstants.setSpaceBelow(style, 15f);
        }

        private JScrollPane getTableComponent() {
                JTable table = new JTable(getModel());

                TableColumn col = table.getColumnModel().getColumn(0);
                col.setHeaderRenderer(new MyTableHeaderRenderer8());

                col = table.getColumnModel().getColumn(1);
                col.setHeaderRenderer(new MyTableHeaderRenderer9());
                table.setEnabled(false);

                Dimension d = table.getPreferredSize();
                d.width = 300;

                table.setPreferredScrollableViewportSize(d);
                return new JScrollPane(table);
        }

        public AbstractTableModel getModel() {

                int t = 0;

                for (int ty = 0; ty < columncount; ty++) {
                        String[] rowsplit = columntext.get(ty).split("\t", -1);

                        for (int tz = 0; tz < rowsplit.length; tz++) {

                                rowceil.add(rowsplit[tz]);

                                rowval.put(String.valueOf(ty) + (tz), rowsplit[tz]);

                        }

                }

                final HashMap<String, Double> map = new HashMap<String,
Double>();
                ValueComparator bvc = new ValueComparator(map);
                final TreeMap<String, Double> sorted_map = new TreeMap(bvc);
```

```
                for (int y = 0; y < columncount; y++) {

                        map.put(rowval.get(String.valueOf(y) + (0)).toString(),
                                        Double.parseDouble(rowval.get(
                                                        String.valueOf(y) +
(1)).toString()));

                }

                // System.out.println("unsorted map");
                for (String key : map.keySet()) {

                }

                sorted_map.putAll(map);

                int y1 = 0;

                for (String key : sorted_map.keySet()) {

                        if (sorted_map.get(key) != null) {

                                float p = (float) Math.pow(10, 2);
                                double Rval = sorted_map.get(key) * p;
                                float tmp = Math.round(Rval);

                                String typ = (float) tmp / p + "";

                                ceilval.put(String.valueOf(y1) + (0), key);
                                ceilval.put(String.valueOf(y1) + (1), typ);
                        } else {
                                ceilval.put(String.valueOf(y1) + (0), key);
                                ceilval.put(String.valueOf(y1) + (1), "N/A");
                        }
                        y1 += 1;
                }

                // n
                double[] value = new double[5];
                String[] languages = new String[5];
                // end n

                int y21 = 0;

                for (String key : sorted_map.keySet()) {

                        if (sorted_map.get(key) != null && y21 < 5) {
                                System.out.println("1: "
                                                + ceilval.get(String.valueOf(y21) +
(0))
                                                                .toString());
                                System.out.println("2: "
                                                + ceilval.get(String.valueOf(y21) +
(1))
                                                                .toString());
                                value[y21] = Double.parseDouble(ceilval.get(
```

```
                                                        String.valueOf(y21) +
(1)).toString());
                                                languages[y21] = ceilval.get(
                                                        String.valueOf(y21) +
(0)).toString();
                                        }
                                        y21 += 1;
                                }

                                textArea.add(new SimpleBarChart(value, languages,
                                                "Programming Languages"));

                                if (columncount <= 5) {
                                        return new AbstractTableModel() {

                                                public int getColumnCount() {
                                                        return 2;
                                                }

                                                public int getRowCount() {
                                                        return columncount;
                                                }

                                                public Object getValueAt(int row, int col) {
                                                        return ceilval.get(String.valueOf(row) +
(col));

                                                }
                                        };
                                } else {
                                        return new AbstractTableModel() {

                                                public int getColumnCount() {
                                                        return 2;
                                                }

                                                public int getRowCount() {
                                                        return 5;
                                                }

                                                public Object getValueAt(int row, int col) {

                                                        return ceilval.get(String.valueOf(row) +
(col));

                                                }
                                        };

                                }

                        }

                        class ValueComparator implements Comparator {

                                Map base;
```

```
                                        public ValueComparator(Map base) {
                                                this.base = base;
                                        }

                                        public int compare(Object a, Object b) {

                                                if ((Double) base.get(a) < (Double) base.get(b)) {
                                                        return 1;
                                                } else if ((Double) base.get(a) == (Double) base.get(b)) {
                                                        return 0;
                                                } else {
                                                        return -1;
                                                }
                                        }
                                }

                                private void setContent(StyledDocument doc, String text) {
                                        try {
                                                doc.insertString(0, text, doc.getStyle("base"));
                                                doc.insertString(doc.getLength(), "\n",
                                                                doc.getStyle("table"));
                                        } catch (BadLocationException e) {
                                        }
                                }

                                private void styleContent(StyledDocument doc) {
                                        Style style = doc.getStyle("base");
                                        doc.setLogicalStyle(0, style);
                                }

                        });
                        jTextArea1.setText(tempQ1);

                }
        }

        private void jButton2_actionPerformed(ActionEvent e) {
                jTextField1.setText("");
                jTextArea1.setText("");

                jButton1.setEnabled(true);

                jCheckBox1.setSelected(false);
                jCheckBox2.setSelected(false);

                bgroup.clearSelection();

                check1 = false;
                check2 = false;
        }

        public static void main(String[] args) throws IOException {
                JFrame.setDefaultLookAndFeelDecorated(true);
                JDialog.setDefaultLookAndFeelDecorated(true);
                try {
                        UIManager
```

```
        .setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

                } catch (Exception ex) {
                }

                IndexWriter w = createIndex.createWriter();

                File f = new File("symptoms_info.txt");
                System.out.println(f.getAbsolutePath());

                createIndex.buildIndex(w, "Symptoms", f.getAbsolutePath());

                File f1 = new File("treatments_info.txt");
                System.out.println(f1.getAbsolutePath());

                createIndex.buildIndex(w, "Drugs", f1.getAbsolutePath());

                createIndex.stopWriter(w);

                new Health();
        }

}
```

createIndex.java

```
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.queryParser.ParseException;
import org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopScoreDocCollector;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.LockObtainFailedException;
import org.apache.lucene.store.RAMDirectory;
import org.apache.lucene.util.Version;

import de.l3s.boilerpipe.extractors.CanolaExtractor;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
/**
 *
 * @author User
 */
public class createIndex {

   // public static StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_CURRENT);
            public static StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_29);
            static Directory index = new RAMDirectory();

    public static void addDoc(IndexWriter w, String value, String fieldname) throws IOException {
        Document doc = new Document();
        doc.add(new Field(fieldname, value, Field.Store.YES, Field.Index.ANALYZED));
        w.addDocument(doc);
    }

  public static IndexWriter createWriter() throws CorruptIndexException, LockObtainFailedException, IOException
  {
                IndexWriter w = new IndexWriter("indexes", analyzer, true,
IndexWriter.MaxFieldLength.UNLIMITED);

                return w;
  }

  public static void stopWriter(IndexWriter w) throws CorruptIndexException, LockObtainFailedException,
IOException
  {
                w.close();
  }


    public static void buildIndex(IndexWriter w, String fieldname, String filename) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader(filename));

         String line = null;
        String q = null;

        while ((line = br.readLine()) != null) {
              q = line.replaceAll("ï»¿", "");
            addDoc(w, q.toLowerCase(), fieldname);
        }
        q = null;
        br.close();
    }


}
```

searchIndex.java

```
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;

import org.apache.lucene.analysis.KeywordAnalyzer;
```

```
import org.apache.lucene.analysis.SimpleAnalyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.CorruptIndexException;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.queryParser.ParseException;
import org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.HitCollector;
import org.apache.lucene.search.Hits;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopScoreDocCollector;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.store.RAMDirectory;
import org.apache.lucene.util.Version;

/**
 *
 * @author User
 */
public class SearchIndex {

            static HashMap hashMap = new HashMap();
    static HashMap hashMap2 = new HashMap();

    static HashMap reason_hash = new HashMap();
    static HashMap side_hash = new HashMap();
    static HashMap sideproblem_hash = new HashMap();
    static HashMap stop_hash = new HashMap();

   static ArrayList<String> myArr = new ArrayList<String>();
    static int ic = 0;
    static StringBuffer stb0 = new StringBuffer();
    static StringBuffer stb1 = new StringBuffer();
    static StringBuffer stb2 = new StringBuffer();
    static StringBuffer stb3 = new StringBuffer();
    static StringBuffer stb4 = new StringBuffer();

    public static StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_CURRENT);
    public static Directory index = new RAMDirectory();

    public static String searchCelebrityname(String type, String querry) throws CorruptIndexException, IOException,
ParseException {

            myArr.clear();

            String strr = "";
       float t1 = 0, t2;
     // StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_CURRENT);
       KeywordAnalyzer analyzer = new KeywordAnalyzer();

        Query query = new QueryParser(Version.LUCENE_CURRENT, type, analyzer).parse(querry);

        File file = new File("indexes");
```

```
        //File file = new File("/home/ec2-user/RssFeedExtraction/indexes");

        IndexReader indexReader = IndexReader.open(FSDirectory.open(file), true);

        IndexSearcher indexSearcher = new IndexSearcher(indexReader);

        TopScoreDocCollector collector = TopScoreDocCollector.create(99999, true);

        indexSearcher.search(query, collector);

       //  Hits hjit = indexSearcher.search(query);

        ScoreDoc[] hitss = collector.topDocs().scoreDocs;

       String[] temp = null;
       String temp1 = "";

   int t;
   int tr;
   int ts;
   int tsp;
   int tsrn;
      // System.out.println("Found " + hitss.length + " hits.");
       if (hitss.length > 0) {

            //new try
            for (int i = 0; i < hitss.length; i++) {

             int docId = hitss[i].doc;

             Document d = indexSearcher.doc(docId);

             strr = d.get(type);

             temp = d.get(type).split(",", -1);

             if(type.equals("Symptoms"))
             {
//                   System.out.println("In Symptoms");
                   if(temp.length == 5)
                   {
                         if(querry.contains(temp[0]))
                         {
                               if(temp[0].length() > 0)
                               {
                                     if( hashMap.containsKey(temp[0]) ){
                                           t = (Integer) hashMap.get(temp[0]);
                                           t = t + Integer.parseInt(temp[3].trim());
                                           hashMap.put(temp[0], t);
                                     }
                                     else if(temp[3].length() > 0){
                                     hashMap.put(temp[0], Integer.parseInt(temp[3].trim().replaceAll("\"",
"")));
                                     }
                               }
                         }
```

```
		}
	}
	else if(type.equals("Drugs"))
	{
//		System.out.println("In Drugs");
		 if(temp.length == 12)
		{
			if(querry.contains(temp[0]))
			{

				if(temp[0].length() > 0)
				{

					// for reasons_taken_number
					if( reason_hash.containsKey(temp[0]) )
					{
						tr = (Integer) reason_hash.get(temp[0]);
						if(temp[5].length() > 0)
						{
						//System.out.println("temp[5]: "+temp[5]);
							tr = tr +
Integer.parseInt(temp[5].trim().replaceAll("\"", ""));
							//System.out.println("reason_hash.put(temp[0], tr):
"+temp[0]+", tr: "+tr);
							reason_hash.put(temp[0], tr);
						}
					}
					else if(temp[5].length() > 0)
					{
						//System.out.println("reason_hash.put(temp[0],
Integer(temp[5])): "+temp[0]+", tr: "+Integer.parseInt(temp[5].trim().replaceAll("\"", "")));
						reason_hash.put(temp[0],
Integer.parseInt(temp[5].trim().replaceAll("\"", "")));
					}

					//for side_effects_number
					if( side_hash.containsKey(temp[0]) )
					{
						ts = (Integer) side_hash.get(temp[0]);
						if(temp[7].length() > 0)
						{
						//System.out.println("temp[5]: "+temp[5]);
							ts = ts +
Integer.parseInt(temp[7].trim().replaceAll("\"", ""));

							//System.out.println("side_hash.put(temp[0], ts):
"+temp[0]+", ts: "+ts);
							side_hash.put(temp[0], ts);
						}
					}
					else if(temp[7].length() > 0)
					{
						//System.out.println("side_hash.put(temp[0],
Integer(temp[7])): "+temp[0]+", tr: "+Integer.parseInt(temp[7].trim().replaceAll("\"", "")));
						side_hash.put(temp[0],
Integer.parseInt(temp[7].trim().replaceAll("\"", "")));
```

```
                                        }

                                        //for side_effects_problem_number
                                        if( sideproblem_hash.containsKey(temp[0]) )
                                        {
                                                tsp = (Integer) sideproblem_hash.get(temp[0]);
                                                if(temp[9].length() > 0)
                                                {
                                                //System.out.println("temp[5]: "+temp[5]);
                                                        tsp = tsp +
Integer.parseInt(temp[9].trim().replaceAll("\"", ""));

                                                        //System.out.println("sideproblem_hash.put(temp[0],
tsp): "+temp[0]+", tsp: "+tsp);
                                                        sideproblem_hash.put(temp[0], tsp);
                                                }
                                        }
                                        else if(temp[9].length() > 0)
                                        {
                                                //System.out.println("sideproblem_hash.put(temp[0],
Integer(temp[9])): "+temp[0]+", tsp: "+Integer.parseInt(temp[9].trim().replaceAll("\"", "")));
                                                sideproblem_hash.put(temp[0],
Integer.parseInt(temp[9].trim().replaceAll("\"", "")));
                                        }

                                        //for stop reasons
                                        if(stop_hash.containsKey(temp[0]) )
                                        {
//                                              System.out.println("temp[0]: "+temp[0]);
//                                              System.out.println("12: "+reason_hash.get(temp[0]));
                                                tsrn = (Integer) stop_hash.get(temp[0]);
                                                if(temp[11].length() > 0)
                                                {
                                                //System.out.println("temp[5]: "+temp[5]);
                                                        tsrn = tsrn +
Integer.parseInt(temp[11].trim().replaceAll("\"", ""));

                                                        //System.out.println("stop_hash.put(temp[0], tsrn):
"+temp[0]+", tsrn: "+tsrn);
                                                        stop_hash.put(temp[0], tsrn);
                                                }
                                        }
                                        else if(temp[11].length() > 0)
                                        {
                                                //System.out.println("stop_hash.put(temp[0],
Integer(temp[11])): "+temp[0]+", tsrn: "+Integer.parseInt(temp[11].trim().replaceAll("\"", "")));
                                                stop_hash.put(temp[0],
Integer.parseInt(temp[11].trim().replaceAll("\"", "")));
                                        }

                                }


                        }
                }
        }
```

```
            }


             //System.out.println("Retrieving all values for Number of Patients");
           Iterator iterator = hashMap.entrySet().iterator();

            while(iterator. hasNext()){

              iterator.next();
                        //System.out.println(iterator.next());
            }

          //  System.out.println("Retrieving all values for Reasons Taken");
            iterator = hashMap2.entrySet().iterator();

             while(iterator. hasNext()){
           iterator.next();
                        //System.out.println(iterator.next());
             }

             //end new try



           for (int i = 0; i < hitss.length; i++) {
              int docId = hitss[i].doc;

              Document d = indexSearcher.doc(docId);

              strr = d.get(type);

              temp = d.get(type).split(",", -1);


              if(type.equals("Symptoms"))
              {


                        if(temp.length == 5)
                        {
                                  if(querry.contains(temp[0]))
                                  {
                                            ic+=1;

                        //          int index = querry.indexOf(temp[0]);

//                                            myArr.add(strr);
//                        myArr.add("##");
//

                                  //System.out.println("Index of "+temp[0]+" :"+index+" and its length is:
"+temp[0].length());
//                                            if(temp[0].length() > 0)
//                                            {
//                                                      if( hashMap.containsKey(temp[0]) ){
```

```
//                                                      t = (Integer) hashMap.get(temp[0]);
//                                                      t = t + Integer.parseInt(temp[3].trim());
//                                                      hashMap.put(temp[0], t);
//                                              }
//                                              else if(temp[3].length() > 0){
//                                              hashMap.put(temp[0], Integer.parseInt(temp[3].trim().replaceAll("\"",
""")));
//                                              }
//                                      }
                                        if(temp[0].length() > 0 && temp[1].length() > 0 && temp[2].length() > 0 &&
temp[3].length() > 0  && temp[4].length() > 0 && hashMap.get(temp[0]) != null)
                                        {
                                                myArr.add(strr);
                        myArr.add("##");


                                                //      System.out.println("Value for Symptom: "+temp[0]+"
:"+hashMap.get(temp[0]).toString());
                                                //stb0.append(temp[0]);
                                                //System.out.println("Symptoms: "+temp[0]);
                                        }

                                        if(temp[1].length() > 0)
                                        {
                                        //      stb1.append(temp[1]);
                                //              System.out.println("Number of Patients: "+temp[1]);
                                        }
                                        if(temp[2].length() > 0)
                                        {
                                        //      System.out.println("Description: "+temp[2]);
                                        }
                                        if(temp[4].length() > 0)
                                                //System.out.println("What Patients take for "+temp[0]+": "+temp[4]);


                                        if(temp[3].length() > 0)
                                        {
                                                //(Integer.parseInt(temp[3]) /
Integer.parseInt(hashMap.get(temp[0]).toString()) ) * 100;
                                                 t2 = (Integer.parseInt(hashMap.get(temp[0]).toString()));

                                                t1 = ((Integer.parseInt(temp[3])) * 100) / t2;

                                //              System.out.println("Percentage of Patients taking "+temp[4]+" for
treating "+temp[0]+": "+t1);
                                        }




                                }
                        //myArr.add
                                //System.out.println("In length == 5");
                        }
```

```
		}
		else if(type.equals("Drugs"))
		{
			//if(temp.length == 2)
			//{
				//if(querry.contains(temp[0]))
				//{
				//System.out.println("Drug Name: "+temp[0]);
			//		System.out.println("Category: "+temp[1]);
				//}
		//	}
			StringBuffer drug = new StringBuffer();
			 if(temp.length == 12)
			{
				if(querry.contains(temp[0]))
				{


			drug.delete(0, drug.length());
//					if(temp[0].length() > 0)
//					{
//						if( hashMap2.containsKey(temp[0]) ){
//							t = (Integer) hashMap2.get(temp[0]);
//							if(temp[5].length() > 0)
//							{
//							//System.out.println("temp[5]: "+temp[5]);
//							t = t + Integer.parseInt(temp[5].trim().replaceAll("\"", ""));
//							hashMap2.put(temp[0], t);
//							}
//						}
//						else if(temp[5].length() > 0){
//						hashMap2.put(temp[0], Integer.parseInt(temp[5].trim().replaceAll("\"",
""")));
//						}
//					}

					//System.out.println("L: "+temp.length);
					if(temp[0].length() > 0 )
					{
						//System.out.println("Value for Drug: "+temp[0]+"
:"+hashMap2.get(temp[0]).toString());
						//System.out.println("Drug Name: "+temp[0]);

						drug.append(temp[0]+",");
					}
					if(temp[1].length() > 0 )
					{
						//System.out.println("Category: "+temp[1]);

						drug.append(temp[1]+",");
					}
					if(temp[2].length() > 0 )
						{
						drug.append(temp[2]+",");
						//System.out.println("Number of Patients: "+temp[2]);
						}
```

```
                        if(temp[3].length() > 0 )
                        {
                                drug.append(temp[3]+",");
                                //System.out.println("Description: "+temp[3]);
                        }
                        if(temp[4].length() > 0 )
                        {
                                drug.append(temp[4]+",");
                                //System.out.println("Reasons Taking "+temp[0]+" : "+temp[4]);
                        }
                        if(temp[5].length() > 0 )
                        {
                                drug.append(temp[5]+",");
                                //System.out.println("Number of Patients Reporting Reasons:
"+temp[5]);
                        }


                        if(temp[6].length() > 0 )
                        {
                                drug.append(temp[6]+",");
                                //System.out.println("Side Effects: "+temp[6]);
                        }

                        if(temp[7].length() > 0 )
                        {
                                drug.append(temp[7]+",");
                                //System.out.println("Number of Patients Reporting Side effects:
"+temp[7]);
                        }
                        if(temp[8].length() > 0 )
                        {
                                drug.append(temp[8]+",");
                                //System.out.println("Side Effect Problem Severeties: "+temp[8]);
                        }

                        if(temp[9].length() > 0 )
                        {
                                drug.append(temp[9]+",");
                                //System.out.println("Number of Patients Reporting Side Effect
Problem Severeties: "+temp[9]);
                        }

                        if(temp[10].length() > 0 )
                                {
                                drug.append(temp[10]+",");
                                //System.out.println("Stop Reasons: "+temp[10]);
                                }

                        if(temp[11].length() > 0 )
                        {
                                drug.append(temp[11]+",");
                                //System.out.println("Number of Patients Reporting Stop Reasons:
"+temp[11]);
                        }
```

```
                                        //System.out.println("strr in drugs, ==12, contains: "+strr);

                                        myArr.add(strr);
                        myArr.add("##");
                                }
                        }



                }


            }



        }


        indexSearcher.close();
        indexReader.close();

        if(myArr.size() == 0)
                        {
                        }
                        else{
                                //System.out.println("ic: "+ic);
//                      System.out.println("myArr.size() 1: "+myArr.size());
//                      System.out.println("myArr.toString() 1: "+myArr.toString());
                        return(myArr.toString());
                        }
//       System.out.println("myArr.size() 2: "+myArr.size());
//                      System.out.println("myArr.toString() 2: "+myArr.toString());

        return(myArr.toString());
        //return "";
    }
}
```

MyTableHeaderRenderer.java

```
import java.awt.Component;

import javax.swing.JLabel;
import javax.swing.JTable;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableCellRenderer;


public class MyTableHeaderRenderer extends JLabel implements TableCellRenderer {
    // This method is called each time a column header
    // using this renderer needs to be rendered.
    public Component getTableCellRendererComponent(JTable table, Object value,
            boolean isSelected, boolean hasFocus, int rowIndex, int vColIndex) {
        // 'value' is column header value of column 'vColIndex'
```

```
        // rowIndex is always -1
        // isSelected is always false
        // hasFocus is always false

        // Configure the component with the specified value
        setText("Reasons for Taking the Drug");

        // Set tool tip if desired
        setToolTipText("Reasons for Taking this Drug");

        // Since the renderer is a component, return itself
        return this;
    }

    // The following methods override the defaults for performance reasons
    public void validate() {}
    public void revalidate() {}
    protected void firePropertyChange(String propertyName, Object oldValue, Object newValue) {}
    public void firePropertyChange(String propertyName, boolean oldValue, boolean newValue) {}
}
```

MyTableHeaderRenderer1.java

```
import java.awt.Component;

import javax.swing.JLabel;
import javax.swing.JTable;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableCellRenderer;


public class MyTableHeaderRenderer3 extends JLabel implements TableCellRenderer {
    // This method is called each time a column header
    // using this renderer needs to be rendered.
    public Component getTableCellRendererComponent(JTable table, Object value,
            boolean isSelected, boolean hasFocus, int rowIndex, int vColIndex) {
        // 'value' is column header value of column 'vColIndex'
        // rowIndex is always -1
        // isSelected is always false
        // hasFocus is always false

        // Configure the component with the specified value
        setText("Percentage of Patients having Side Effects");

        // Set tool tip if desired
        setToolTipText((String)value);

        // Since the renderer is a component, return itself
        return this;
    }

    // The following methods override the defaults for performance reasons
    public void validate() {}
    public void revalidate() {}
    protected void firePropertyChange(String propertyName, Object oldValue, Object newValue) {}
    public void firePropertyChange(String propertyName, boolean oldValue, boolean newValue) {}
}
```