Mining Healthcare Forums

Akash Nigam †  
anigam6@asu.edu

Sandeep Nadella †  
vnadell1@asu.edu Mayur Padaval †  
mpadaval@asu.edu

Shubham Gupta †  
sgupt198@asu.edu Nitika Garg †  
nnitika@asu.edu

Vinay Matcha †  
vmatcha@asu.edu

† *Graduate Student*, Computer Science (MCS), Arizona State University

*Abstract*—Healthcare forums contain tons of information about diseases, treatments, and drugs especially from the personal experience of people. But that information is largely in unstructured form i.e. in the natural language as people recount their experiences.

The lack of information in a structured form regarding disease-symptom-drug relation is one of the major drawbacks when it comes to drawing useful inferences and insight from data. This is the area which we are trying to address by mining healthcare websites. If large amounts of data on the medical information of various patients are analyzed and adequately interpreted, it can be used to identify the medical trends. This, in turn, can improve the quality of the pharmaceutical industry's decision-making process.

Keywords—Healthcare Mining web data mining, Web scraping, Web Mining, MetaMap, Ontology, Healthcare forums, UMLS

# Introduction

The aim of this project is to build an information extraction system that can turn unstructured medical healthcare data from user posts of multiple sources (in WebMD.com[6], Drugs.com[7], Patient.info[8]) into structured information and build a parametric search interface for a category (a disease/symptom/drug). Using the search interface, users will be able to get the different attributes of input disease, possible diseases for given symptoms, or the usage of the input drug, etc. Also, the search output data attributes will be ranked based on their occurrence frequency on data sources, to help users in differentiating the more common ones to the less frequent ones. The attributes of a disease include symptoms, the category of disease, treatment.

# Problem Statement

Our work is inspired by the following existing problems:

## Unstructured Health-related experience data

Healthcare forums allow a user to write their healthcare experience without following any particular format. It allows users to recount and express their experience in a more natural way hence increase the usability of the platform and makes it human readable. But this flexibility reduces the ease of extracting knowledge out of that data using machines as natural language is largely unstructured.

## Experiences spread across Multiple Sources

There are multiple forums on the web today that extracting and combining knowledge from data on a large scale can be very helpful in finding trends.

## No Disease - Symptom search tools for data mined from user posts

Due to the lack of structure, there is no easy way to get the aggregated data from all the content from multiple sources in one place.

Following is the solution Proposed:

## Unstructured Data ➔ Structured Data

Our goal is to mine data from these forums and convert it into a structured form.

|  |  |
| --- | --- |
| **Unstructured Data**  *“I saw an orthopedist because of pain and swelling in both knees 2 years ago. Arthritis Org said I need a total knee replacements. I am 75 and do not want to undergo surgery so I curtailed hiking which had been causing pain, I lost some*  *weight. They don't hurt any*  *worse since starting this*  *exercise.”* | **Structured Data**  *Disease*:  Arthritis  *Symptoms*:  Pain NOS Adverse Event, Swelling, Weight decreased,  *Treatment*:  Knee Replacement Arthroplasty, Exercise |

Table 1. Structured data retrieved from Unstructured data

## Search Interface to query Top diseases and symptoms

By using the data collected from the healthcare forums, it can be aggregated in database and a search interface can be created to query top symptom, treatments for a disease and vice versa.

|  |  |  |
| --- | --- | --- |
| **Top Diseases for a Symptom**  Symptom: Sore Throat  **580** people reported Sore Throat.   * **70%** had Epiglottitis * **64%** had Influenza | **Top Symptoms of a Disease**  Disease: Malaria  **980** people reported Malaria.   * **80%** reported Fever * **70%** reported Headache | **Top Treatment for a Disease**  Disease: Influenza  **1193** people reported influenza   * **93%** took Rapivab * **80%** took Relenza |

Table 2. Search output

# Related Work

A variety of state-of-the-art methods have been employed for information extraction in humongous data present in medical and health domain. As clinical notes which make up the most significant component of the Electronic health records are mostly unorganized

Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

## SympGraph: A framework for mining clinical notes through symptom relation graph

One of the excellent works is SympGraph [1] mining framework which can be used to model and analyze the symptom relationships using clinical notes. SympGraph is a type of graph in which nodes are symptoms of different diseases and health problems and edges are the relationship between those symptoms. This SympGraph framework can be used to expand some given symptoms by utilizing the created SympGraph.

## MedEx: A medication information extraction system for clinical narratives

Another such admirable work is MedEx [2] which is a natural language processing system that uses clinical notes to extract medication information. It can be used to reduce medication errors when a patient is transferred from one care setting to another. It can be also be used in EMR based clinical research which requires detailed medical information.

# System Architecture and algorithm

Our project is divided into four major parts.

1. Web Scraping
2. Annotating and extracting medical concepts from Text
3. Ontology Creation
4. Search Interface

## Web Scraping

We are scraping multiple healthcare websites. On these websites, the user posts are categorized based on the disease family.  We are extracting following details from each user post -

* Disease name
* Post link
* Post heading
* Post content
* Post tags

Post tags and disease name can be useful for the user posts which does not produce any semantic type for the disease after passing it through MetaMap annotator. If this is the case, we append the disease name and post tags into the post heading and rerun the process.

For scraping, we are using Python’s Scrapy[3] which is a fast and efficient framework that have tools to manage every stage of a web crawl, like Request Manage, Selectors and Pipelines. It uses XPath and CSS selector to identify elements needed on the web page.

Following is the screenshot for the generated CSV

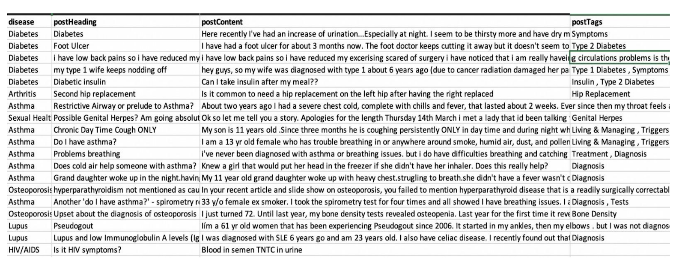


Figure 1. sample csv generated by scrapper

## Annotating and extracting medical concepts from Text

After scraping the data from healthcare forums, we have the user comments in a CSV file. Now, we need to identify diseases, symptoms and treatments from the post. There are many different terminologies for a single disease and symptom. Different people use different terms to describe a disease. So, we need to have the knowledge of a medical domain expert who identify these different terminologies as a single disease, symptom or treatment.

**UMLS**: UMLS is a medical Thesaurus developed by NLM (National Library of Medicine). It consists of multiple standardized set of medical terminologies and relation among them.

It has three knowledge sources.

|  |  |  |
| --- | --- | --- |
| **Metathesaurus**  1 million+  **biomedical**  **concepts** from over  100 sources | **Semantic Network**  135 broad **categories**  and 54 **relationships**  between categories | **SPECIALIST Lexicon & Tools**  **lexical information** and **programs** for language  processing |

Table 3. Components of UMLS

Metathesaurus handles synonyms and assigns unique IDs. Each term is called a “concept” and it also maintains hierarchical relations among concepts.

Semantic network categorizes concepts among diseases, symptoms, treatment, anatomy etc.

Specialist Lexicon tools handle the NLP part and spelling variants etc.

**MetaMap:** Now, we have this expert knowledge from UMLS, we need a tool that can use this knowledge to extract and annotate data. MetaMap is one such tool that annotates and extract medical concepts from text using UMLS. It is also developed by NLM. It is highly configurable in terms of selecting UMLS resources and output etc. It uses NLP and computational-linguistic techniques

A sample text processed by MetaMap is as follows

|  |  |
| --- | --- |
| **Input Text**  *“I saw an orthopedist*  *because of pain and swelling*  *in both knees 2 years ago.*  *Arthritis Org said I need*  *total knee replacements. I*  *am 75 and do not want to*  *undergo surgery so I*  *curtailed hiking which had*  *been causing pain, I lost*  *some weight. They don't*  *hurt any worse since starting*  *this exercise.”* | **MetaMap Output**  Pain NOS (Pain NOS Adverse Event) [Finding]  SWELLING (Swelling) [Finding]  knees (Knee) [Body Part, Organ, or Organ Component]  total knee replacements (Knee Replacement Arthroplasty  (procedure)) [Therapeutic or Preventive Procedure]  lost weight (Weight decreased) [Finding]  ARTHRITIS (Arthritis) [Disease or Syndrome]  Exercise (Exercise Pain Management) [Therapeutic or Preventive  Procedure] |

Table 4. Output of MetaMap for the given input text

**MetaMap Annotator Implementation:** This project uses MetaMap annotator Java API for annotating the web data scrapped by the scraper project. The mmserver should be setup and running prior to running this project. The location of the CSV file containing the scraped data, MetaMap options should be given in the class file.

MetaMap’s Java API and mmserver files can be downloaded from MetaM­ap’s official website.

MetaMap’s Java API is highly configurable. We can provide a list of words to ignore, words to include etc. We can also configure the sources it refers from UMLS.

We can also configure the output i.e. which semantic types to retrieve in the output.

For our project, we extracted the following semantic types

1. Diseases i.e. dsyn | Disease or Syndrome,
2. Symptoms i.e. sosy | Sign or Symptom,
3. Treatments i.e. topp | Therapeutic or Preventive Procedure,
4. Drugs i.e. clnd | Clinical Drug and
5. Body part i.e. bpoc | Body Part, Organ, or Organ Component

MetaMap annotates data post by post and write the result into CSV file. To handle posts without a disease name, we got the concept name from the post category under which user posted. The posts which do not contain any useful information will automatically be ignored by MetaMap as nothing would be annotated under given semantic types.

## Ontology Creation

Ontology describes the various entities of our information extraction system and their underlying relationships. The various entities are disease, symptom, Treatment & anatomy.

● We have created Disease, Symptom and Treatment ontology.

● Objects and relationships will be stored using tables in relational DB. We have used MySQL for out project.

● Relationships will consist of disease related with their symptom, treatment, anatomy. These will be used for giving search interface query output.

● The output of MetaMap is inserted into the ontology database. For this we are first loading the MetaMap output csv data into a table and then using SQL stored procedure  and cursor to put that data in our ontology tables.

● Weight of each symptom, treatment, for a particular disease will be kept. This will allow to get k top frequent symptom, treatment for a disease and vice versa.

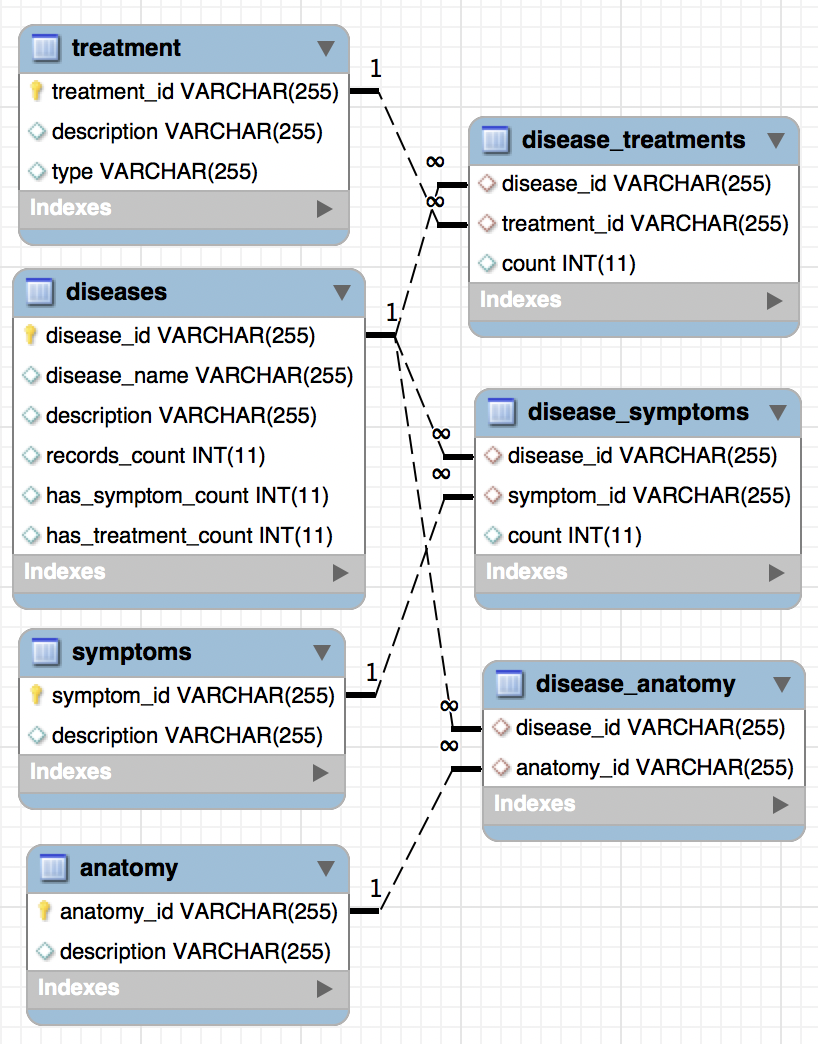


Figure 2. Ontology schema design

Below is the sample database figure showing an example ‘Acne’ disease and its related symptoms. Symptom count denotes the number of users reporting that symptom.

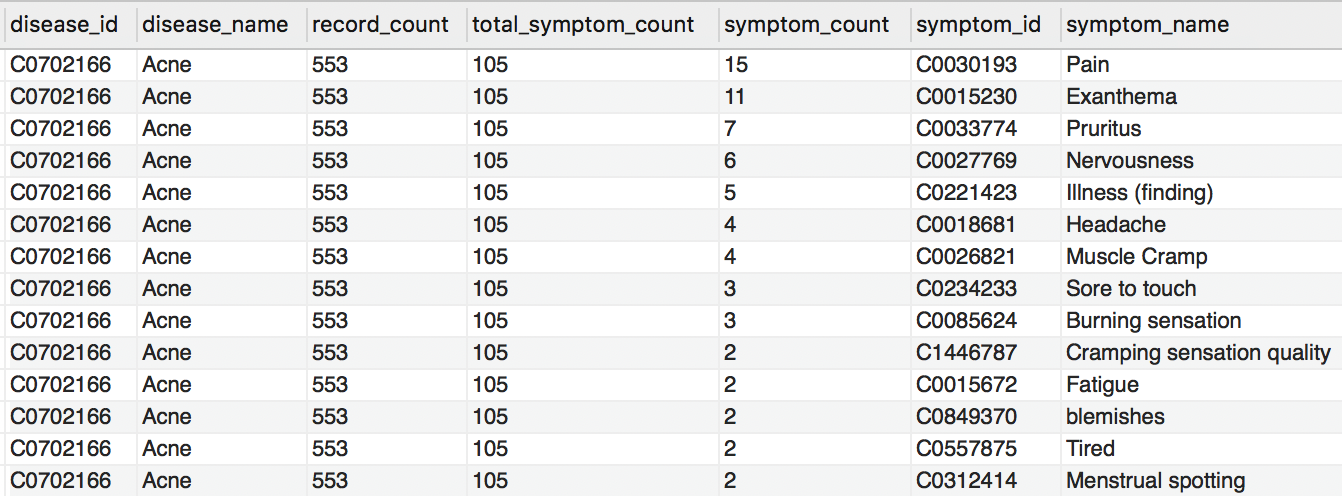


Figure 3. SQL table representing disease-symptom relationship

## Search Interface :

We have created web interface for symptom checker and disease checker. Symptom checker is for checking the diseases associated with symptoms, based on the symptom given by the user, the response will be the diseases that associated with the symptom and percentage of people that reported that symptom. This is like inverse document frequency, percentage of people reported the symptom for a disease will be the total number of people reported for that disease divided by the total number of people reported for all the diseases.

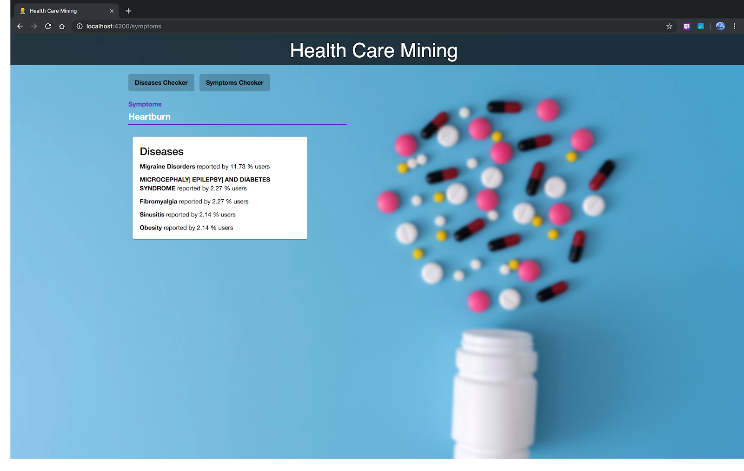


Figure 4. Search interface: Symptom Checker

Disease checker is for checking the symptoms and treatments associated with that disease, (i.e.) given a disease name by the user, the response will be the top five symptoms and treatments associated with that disease.

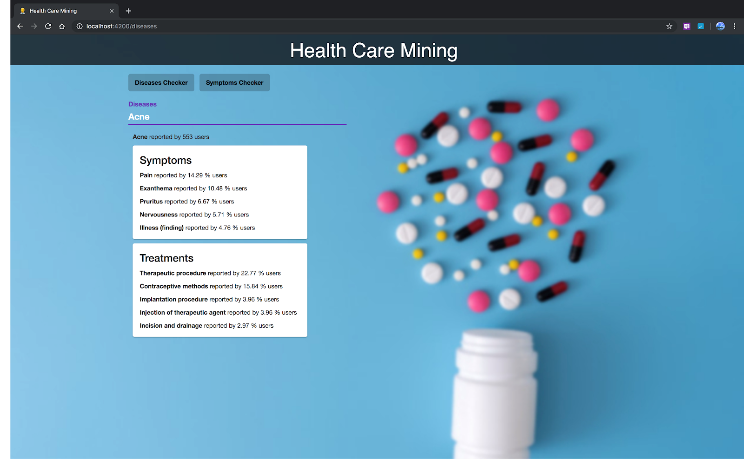


Figure 5. Search Interface: Disease Checker

We have used Angular with Spring Boot for building the search interface. In order to avoid the search calls for the diseases and symptoms which are not in database, we have used ajax query to provide options for diseases and symptoms which are present in database only. Users cannot select the disease or symptoms which are not in database. When the user enters the disease or symptom name in the search interface and submits the button, the REST API call happen and brings the response for the query from the database. We have used JPA hibernate queries for bringing information from the database for the REST API calls.

# Datasets (Descriptions, Sizes and Preprocessing Steps)

The datasets for this project are the Discussion Forums from health care websites where data is posted in unstructured form by users. e.g.

▪ https://www.patient.info/

▪ https://www.drugs.com/

▪ https://www.webmd.com/

We have scraped around 35000 user posts from drugs.com and 10000 from webmd.com and 9000 from patient.info for all available medical conditions on these websites.

# Deployment and Evaluation

The execution pipeline involves running the following modules in sequence

# Scraper

# MetaMap Annotator

# Ontology Creation

# Search Interface Backend Deployment

# Search Interface Frontend Deployment

## Scraper:

The scraper can be run on each site using the following commands from the */HealthCareMining/scraper* folder

$ scrapy crawl patient\_info

$ scrapy crawl webmd

$ scrapy crawl drugs

The debug output of the scraper is as follows

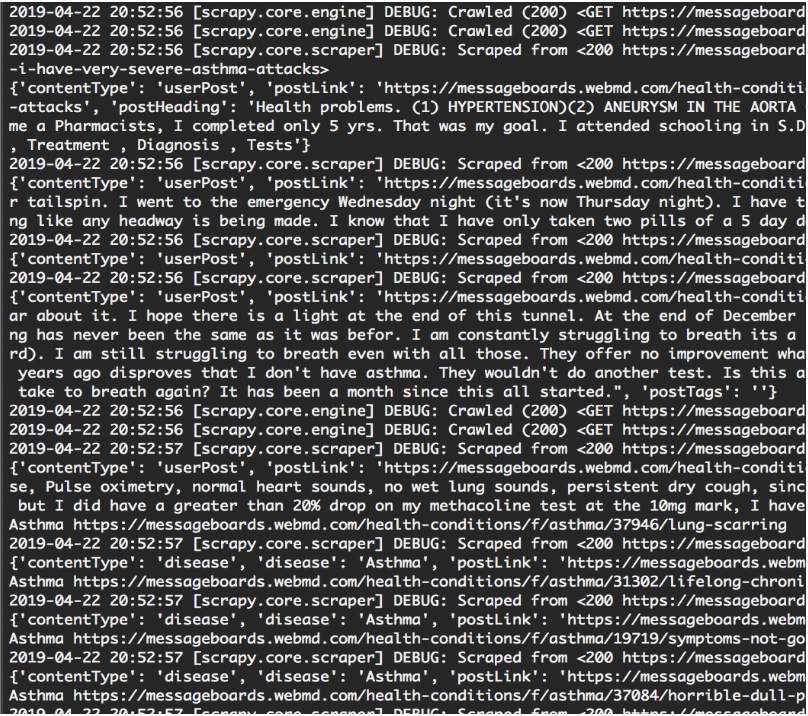
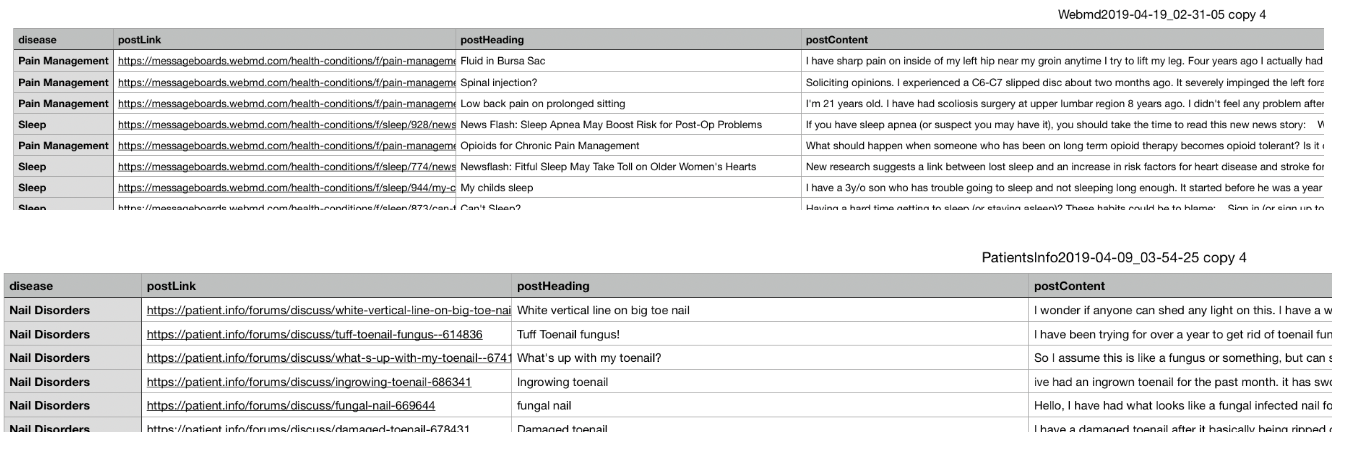


Figure 6. Debug output of the scrapper

As can be seen from the screenshot the scraper crawls through the various post links and extracts the post content from the HTML tags. The *contentType* represents the type of data being extracted. It is then used for mapping the post category name with the post content. Various content types like *postHeadings, postLink, postTags, postContent* are extracted.

The extracted data is written to a CSV file under */HealthCareMining/scraper/data* folder. The structure of the output of scraper if as follows



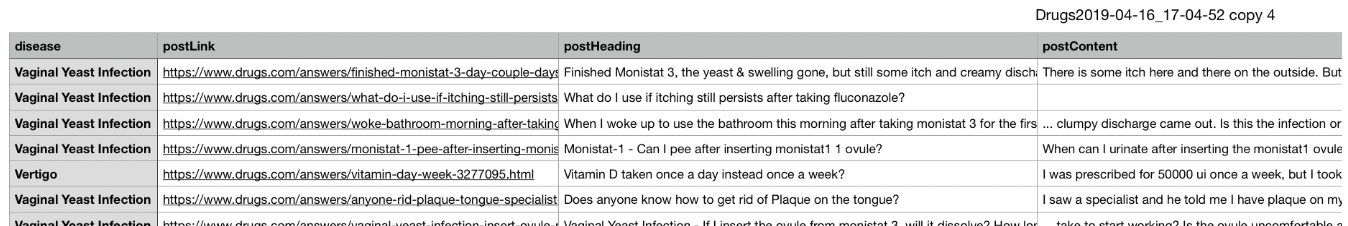


Figure 7. Sample CSV generated by scraping WebMD, patientinfo, drugs.com

* 1. *MetaMapAnnotator:*

These files will be fed as input to the next module in the pipeline which is MetaMap Annotator.

To run MetaMapAnnotator module the MetaMap server pipeline should be setup which includes MedPost/SKR POS Tagger server and Word Sense Disambiguation server to be setup and running. The detailed instructions for setup are available under README.md of the MetaMap Annotator module [4].

The debug output is as follows

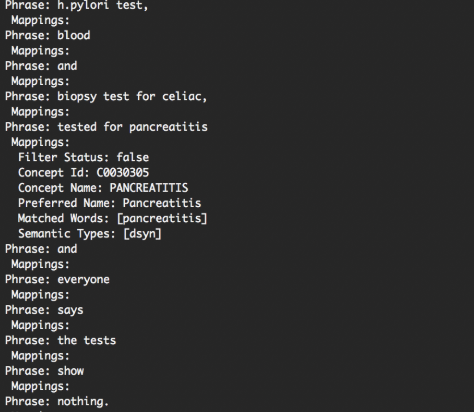


Figure 8. MetaMap debug output

Each user post content is fed to the MetaMap server. The MetaMap server breaks the sentence to phrases and the annotates the lexical match word with the POS tags by feeding it to POS Tagger server. The output is then checked for UMLS concepts and the mappings are generated based on the mappings with highest candidate scores. Here [dsyn] represents disease or syndrome semantic class. The stop words are fed in the *IgnoredWords.csv* file under resources folder which will be filtered out. Helper functions for filtering concepts based on the POS tags are also added and were experimented with.



Figure 9. Stop word list

The concepts identified are written to output CSV file at */HealthCareMining/MetaMapAnnotator/resources* location. The output files are as follows

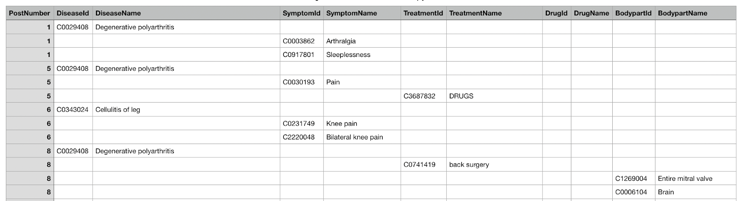


Figure 10. Output of MetaMap

This file is fed as input to the SQL PROC under the Ontology module. The instructions are available under [5]. This module creates the ontology schema and inserts the data into the database. The frequency of occurrence of the concepts is calculated and stored in database by this module. The data from this database is fetched by the Search interface backend and served to the front end.

* 1. *Search Interface Backend Deployment*

Open the *searchInterface* module in the *healthcare-data-mining* as a Maven repository, so that all the dependencies in the pom.xml will be downloaded and configured. Start the MySQL server and change the database configuration in *healthcare-data mining/searchInterface/src/main/resources/application.properties*. Once the changes are made, run spring boot application file *searchInterface/src/main/java/com/swm/searchInterface/SearchInterfaceApplication.java*. Apache Tomcat server will be started under URL, http://localhost:8080/.

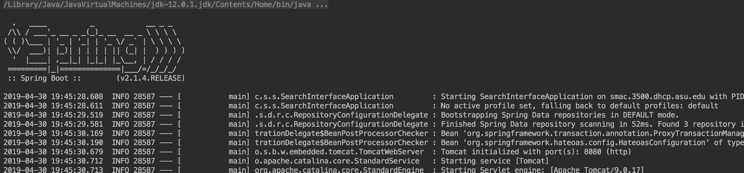


Figure 11. Debug output of back-end server-1

* 1. *Search Interface Frontend Deployment*

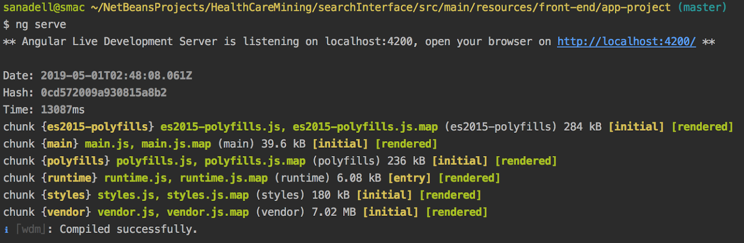
Go to the folder */healthcare-data-mining/searchInterface/src/main/resources/front-end/app-project* in terminal and run the command *ng serve* which will serve the frontend HTML, CSS, JS resources under URL, http://localhost:4200/.  


Figure 12. Debug output of back-end server-2

The search interface looks like this

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Figure 13. Search Interface

# Division of Work and Contributions

|  |  |
| --- | --- |
| **Task** | **Contributors** |
| * Building Web Scraper for scraping unstructured data from multiple sources * Cleaning the user posts for html tags | Akash Nigam, Mayur Padaval |
| * Building Annotator pipeline using MetaMap * Building filters for stop words * Integration of MetaMap with rest of the modules | Sandeep Nadella |
| * Customizing MetaMap to filter required Semantic type from MetaMap output. * Handling posts with missing disease names. | Nitika Garg |
| * Designing the schema for disease, symptom, treatment ontology * Wring queries for schema/table creation. * Writing stored procedure for data insertion in Ontology tables | Shubham Gupta |
| * Setting up Spring and Angular projects * Handling JPA hibernate calls * Handling REST API calls | Vinay Matcha |
| * UI design | Vinay Matcha, Nitika Garg, Mayur Padaval, Sandeep Nadella |
| * Integration and Testing | All |

# Conclusions

Our main goal is to mine unstructured healthcare data from multiple sources and gain some insights. We extracted data from the discussion forum of three of the most popular healthcare websites. We used MetaMap annotator that finds relative semantic types for diseases, symptoms, and treatments for each scraped post.  We designed and implemented the ontology that defines and stores the relationship between different symptoms, diseases, and treatment. Finally, we created a search interface that has disease checker and symptom checker. By using disease checker, the user can find the top 5 symptoms and treatments for the selected input disease. Similarly, the symptom checker returns the top 5 most probable diseases for the input symptom.  We are also providing the percentage of people reporting the particular disease for the symptom and vice versa.

For the sake of simplicity, we assumed that user posts do not contain negative contextual information or have multiples diseases. Also, currently our application supports search operation on only one symptom. We can extend our application to handle multiple symptoms to further narrow down the search for the disease. The project source code is available in GitHub [9].

# References

1. Parikshit Sondhi, Jimeng Sun, Hanghang Tong, ChengXiang Zhai: SympGraph: a framework for mining clinical notes through symptom relation graphs. KDD 2012: 1167-1175
2. Xu H, Stenner SP, Doan S, Johnson KB, Waitman LR, et al. (2010) MedEx: a medication information extraction system for clinical narratives. J Am Med Inform Assoc 17: 19–24
3. <https://docs.scrapy.org/en/latest/intro/overview.html>
4. <https://github.com/akashnigam/healthcare-data-mining/tree/master/MetaMapAnnotator>
5. <https://github.com/akashnigam/healthcare-data-mining/tree/master/ontology>
6. [https://messageboards.webmd.com](https://messageboards.webmd.com/)
7. <https://patient.info/forums>
8. <https://www.drugs.com/answers/conditions>
9. <https://github.com/akashnigam/healthcare-data-mining>