Healthcare Mining
CSE 573 Semantic Web Mining
Project Portfolio, Spring 2020

Manish Seal
Computer Science and Engineering
Arizona State University
Tempe, AZ, USA
mseal1@asu.edu
ASU ID: 1215110442

I. Introduction

Healthcare is an important aspect in our daily life. Healthcare tips can now be found all over the internet from various sources, ranging from doctors doing podcasts and writing blogs to people describing their personal experiences and benefits obtained from medicines/drugs on forums. Now it is all good to have so much of information available, but how much of the data is easily accessible? Can all the data be compiled and provided as a result of a single search? While ways for doing that are still going on and it is not completely perfect. Websites like WebMD provide message boards where users share their experiences on a particular drug or disease. PatientsLikeMe is another such website where users provide a first hand experience of thier experience during a disease. Another website called Healthline provides a list of posts from users based on text similarity matching, giving the most similar results containing the symptoms queried by an user. One can search by providing symptoms to find the related posts. Mayo Clinic is another such website, it contains the information for over a thousand of unique diseases, providing an overview of the disease, the possible symptoms, and the moment when one needs to see a doctor. All of its data is curated by trusted medical professions in one place. So in this project we are creating a system which is compiled from resources from medical websites, forums and message boards. For this we first scrape data from the websites. Secondly, the data in their raw form will be processed to generate tokens and will be indexed into a Solr database for fast retrieval. We also created an application server which does two kinds of search. The first one takes input from user, tokenizes the input,does a query from Solr, which is then refined it using a vector similarity technique and returns the result to the user. The result consists of two lists, one is a list of possible related diseases and more information about them, the other is a list of posts on forums which are similar to the symptoms/text entered by the user. The second type of search the user can do is to search for treatments related to a disease, that one returns a list of reviews available for a certain drug for a disease. The next section describes the various terminologies, the different components, their interactions and the way the whole thing works. In the third section we discuss the results obtained. The fourth and fifth section discusses about my contribution towards the project and the skills I gained from the project respectively.

II. Explanation and Implementation of Solution

We provide the image of the system architecture in 1. Below we explain the the different parts of the solution:

A. Dataset: Web Scraping

We first looked through the different healthcare websites and found the ones which have authentic and trusted information. So we restricted ourselves to three websites about which we provide information below:

- WebMD - Message Boards: We used WebMD to extract the symptoms, diseases, and potential treatments. It has large number of forums posts useful for our data. Around 13k forum posts was crawled. Reviews for around 15k unique drugs have been scraped from WebMD. This is a dynamic website for which we had to write a Node.js application which internally uses Puppeteer to scrap the data once the page has been loaded to a headless browser tab.
- Mayo Clinic - Diseases and Conditions: We crawled Mayo Clinic website to get the diseases and their descriptions, curated by experts and trusted all over the world. Around 1183 health condition pages were crawled. Mayo had a static wepage, so it was easy to scrape it using *requests* and *BeautifulSoup* package of python. Each web page provides an overview, symptoms and when to see a doctor information.
- Patient Info: We crawled the Patient.info website, having active community forums that are extensively dedicated to healthcare-related discussions (including symptoms and diseases). More than around 236K, forum posts has been

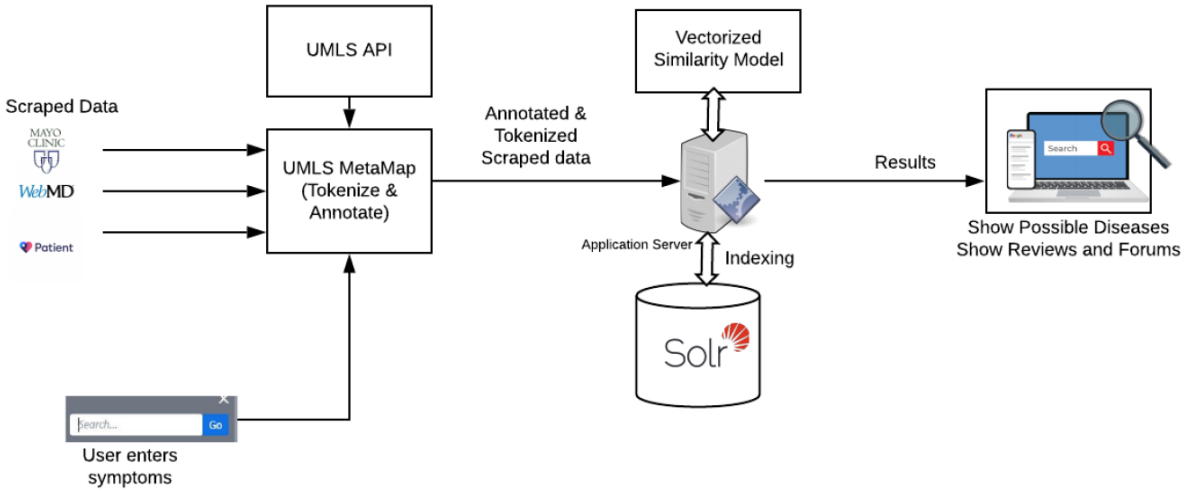

Fig. 1. System Architecture.

crawled. The crawler for the website was developed using Python data scraping framework - Scrapy. We performed the crawling task as a multi-stage process, first links to all the unique forum groups were scraped, then for each such groups links to all the associated forum posts were scraped, finally, the actual content of the forum post and the comments made on the forum posts were scraped/crawled. As a post-processing step, the scraped data were cleaned and stored as a JSON file.

B. Processing the Data

- **Tokenizing using UMLS Metamap API**: The UMLS metamap API [1] provides entity tagging to for the text. It identifies the different words and phrases related to healthcare and provide whether it is a syndrome, a medical condition, time, place, duration or a possible drug name. Using this metamap we can easily find out the entities from a given piece of text and put it inside Solr. Tagging and tokenizing enables faster and relevant retrieval of documents based on time, place, date and other information about the document.
- **Bio-BERT**: Using Bio-BERT [2] on the scraped messages we get a vector for the whole sentence/passage. This embedding is then used to rank the forum posts most similar to the ones entered by the user as returned from Solr.

C. Indexing and Retrieval

- **Indexing**: After obtaining symptoms, diseases, and potential treatments from the crawled entity, we index the entity along with additional metadata which we extract from the forum posts. For indexing, we use Apache Solr as it provides efficient and scalable indexing and searching to and from an inverted index. As we had scraped the needed data beforehand, we index data in batch mode for efficient indexing.
- **Searching**: The user specifies the search query(usually symptom names) in the search box. The user query is then passed through the MetaMap API to extract meaningful entities from the search query. These additional metadata extracted from the user query help us find an appropriate potential disease for the given user query(symptoms).

Both indexing and search requests are handled by the application server - Apache Tomcat.

D. Vector Similarity Measures

The results returned from Solr are based on the tf-idf features of matching. For this reason it fails to capture the context of the document. Now to improve the results, we use Bio-BERT to get an embedding for each document returned from Solr, and also for the query entered by the user. After that we rank each embedding based on a similarity score with respect to the query embedding. For similarity we use cosine similarity. This provides a refinement of the results returned to the user.

E. Application Server

The web application for the search interface is hosted on the Tomcat application server. Two important components of this web application include - the user interface and server request handlers. The search user interface has been implemented using HTML, CSS, JQuery, and JavaScript. The server request handlers are implemented using Java Servlet, which accepts browser client HTTP request(search request) and returns JSON as a response. This JSON response is then used in the client-side to render the search results appropriately.

III. Results

The results which we achieved is that we were able to come up with a search engine where if the user enters a piece of text/sentence, the text is entered into MetaMap to get the medical entities tagged and is then passed on to the application server. It queries Solr to return the top 10 potential diseases that it finds most similar. We also provide option to search for treatments and drugs for a particular disease. The results shown are not recommended by us and it is only provided as reference. The figure 2, shows the sser interface of the search engine, that our team has developed.

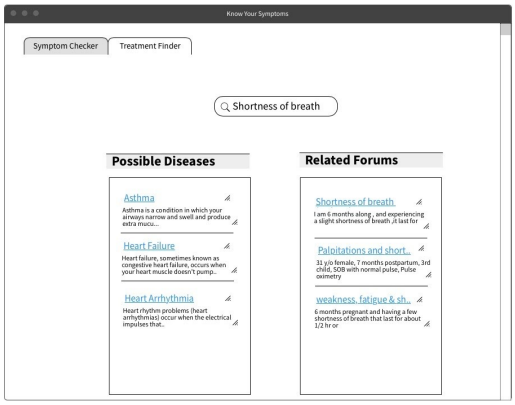

Fig. 2. Application Server Interface

To evaluate the goodness of our query results we will sample a set of symptoms followed by manual testing of the system deliverables. We get the top 10 results from Solr for a query entered by the user. The results are verified manually by a human expert and are identified as TP, TN, FP, FN, based on the top-1 and top-5 score. Top-1 and top-5 score refers to whether the actual disease (ground truth) was present at the top of the list or within the top-5 results returned to the user respectively. Based on the FP and FN, precision and Recall will be calculated and validated. We get the mean of the ratings by the human expert for 30 such queries. The top-1 score was around 56% and the top-5 score was around 67%.

IV. My Contribution towards the Project

In this project there was a lot of work regarding scraping. Me and Manoj were responsible for scraping the Mayo Clinic website which was a static website. Its scraping was pretty straight forward, we figured out the common pattern in the links of websites and wrote a crawler which scraped all the pages of symptoms and diseases within an hour (around 1183 pages were crawled). The code was written in python. I was also responsible for literature study to find out the state-of-the-art ways in healthcare mining and decide viable options for our project. Next I was responsible for setting up the UMLS metamap API, which tokenized and tagged the different medical entities in a given text. Next I also helped in setting up the Bio-Bert script which helped in creating embedding for a document. I also wrote the script which does similarity matching between different embedding for refinement of results returned to the user.

V. Skills and Knowledge Gained

There are quite a few skills I gained from the project. First of all is Scraping, this is the first time I scraped and learned the difference between static and dynamic websites, the one which return the whole data and the other one which returns a java script respectively. Secondly I learned how to convert the semi structured web data into useful structured data using tokenizing and UMLS metamap api. Thirdly I learned about UMLS, it is maintained by National Library of Medicine. UMLS provides medical entity tagging, contains a vast database of medical names, terms, synonyms, diseases and possible symptoms too. Thirdly I learned to use Bio-BERT, how it is used to obtain an embedding of a whole sentence. This is particularly useful and very powerful as it helps to capture the context of a sentence. Fourthly I learned how to design a front end web page which enables user to query.

Also, my team members were an effective support throughout without whom the project is incomplete. My team members were:

- Aditya Deotale
- Anik Pait
- Hemant Singh
- Manoj Kumar Yelljirao
- Varun Chaudhury

We were a team of six for the project who worked best up to their efforts to make the project a success.

References

[1] Bodenreider O. The Unified Medical Language System (UMLS): integrating biomedical terminology. Nucleic Acids Res. 2004 Jan 1;32(Database issue):D267-70. doi: 10.1093/nar/gkh061. PubMed PMID: 14681409; PubMed Central PMCID: PMC308795.

[2] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang, BioBERT: a pre-trained biomedical language representation model for biomedical text mining, Bioinformatics, Volume 36, Issue 4, 15 February 2020