CS F469 Information Retrieval Project - Design Document

Group Members:

ID Number	Name
2017A7PS0075H	Prateek Agarwal
2017A7PS0093H	Shreeya Bharat Nelekar
2017AAPS0409H	Shriya Choudhary
2017AAPS1741H	Shubham Singhal

Medicinal Search Engine

Introduction:

We designed a search engine based on the use of different medicinal drugs for different diseases. The query for the search engine will be the names of diseases and the result will contain files on all such drugs that are used for the treatment of the said diseases. Also, if specified by the user, the search engine can also check that the result does not contain a medicine which has the specified side-effect.
Secondly the search engine would also support a feature where if you query for a particular medicine, the result will display a list of all other medicines that are related i.e. same chemical formula but different brand name, treats the same disease, etc.

Language - Python3

Dataset for the Search Engine:

The corpus of medicinal search engine includes the information of all medicines and drugs available throughout. The information contains the description of the drug, side effects, other brand names, diseases concerned with the medicine etc.

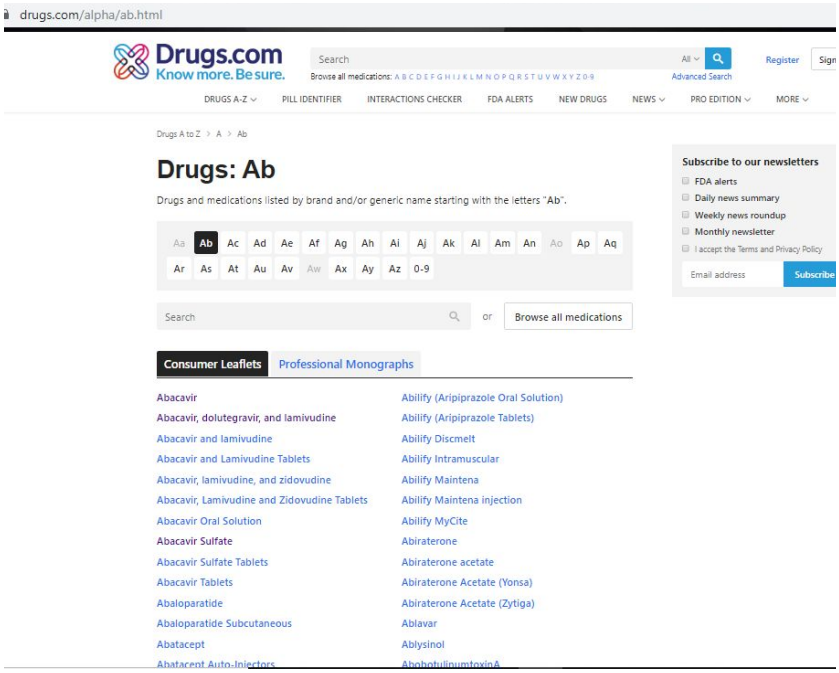
The dataset was extracted by scraping the website - https://www.drugs.com using python libraries.

Web Scraping:

Language - Python3
Libraries - BeautifulSoup
Website used for Scraping - https://www.drugs.com

Stage 1-

Initially all the names of the medicines available on the website was gathered using one script which navigates through the index of all medicines from the website.
URL - https://www.drugs.com/drug_information.html
Medicine Count - **20,850**



All the medicine names and their corresponding URL were stored in a **.txt** file so that it can be used in the 2nd stage of scraping to separately extract the information of each medicine.

Format of **all_medicines.txt** file:
<medicine name 1>=<medicine url 1>
….
….
<medicine name n>=<medicine url n>

Output of Stage 1 of Web Scraping

Stage 2 -

In this stage, the above output file was used to visit the main page of each medicine/drug mentioned in the **all_medicines.txt** file and the required information was extracted.
Medicine information available on the website is segregated into categories as - ‘Overview’, ‘Side Effects’, ‘Dosage’, etc.Hence the entire information available is stored in the form of a **JSON Object** in **MongoDB**.
Dataset Size - **808.4mb**

Data Structures Used:

Dataset
The entire dataset is stored in **MongoDB**, a NoSQL Database.
Each document in mongoDB collection represents one Medicine and the medicine name was used as the primary key for the given document.
The other information was stored as a JSON Object as it would help in easily detecting the different components of medicine information like side effects and description.
This is because we need to ensure the if we search for a particular medication (for ex - cold) then those medicines do not show up in results which include ‘cough’ as one of its side effects. We only need to look for those results where ‘cough’ is present in overview of the medicine

Indexes

The inverted index and term frequency inverse document frequency index are generated by python scripts using nltk library and stored in a form of python ‘dict’ - dictionary.
The dictionary was later stored permanently in a pickle file. Pickle is a library that allows to store a python dictionary contents in a **.p** file and retrieve it back in python as a python dict.

Index Generation:

Before generating an inverted index, we first remove the **stop words** by using stopwords class from nltk package. In this way, we will remove less important data. After removing stop words, we generated tokens using **tokenization**. Tokenization helps in breaking the sequence of characters in keywords and phrases. On this tokenized data, we applied **stemming** algorithm which reduced the tokenized characters into their root words. This process helps in increasing the accuracy of search. The algorithm we used for stemming was Potter’s Stemmer.
In the final step for index generation, we created an inverted index for our stemmed words. In inverted index, we mapped our words with documents which later on will help us to find results for our query.

After inverted index, we used **tf-idf method** for weighting the frequency of our words in documents. Tf-idf stands for term frequency- inverse document and helps in finding the importance of a word in a document.

Search Query:

The search query can be a disease name or a list of symptoms corresponding to which our search engine would return a list of medicinal drugs which can be used for the treatment of the given disease.
The running time for search retrieval is 0.0129 seconds.