Denham Preen

prnden002

Sabelo Xulu

xlxsab001

Information Retrieval of Medical Data

Assignment 2

Introduction

This document describes the process behind building a search engine to find medical information from a dataset of 267 web documents specifically html pages relating to medical illnesses and diseases. The intention of the search engine is to search for symptoms and to ascertain websites on illnesses that are relevant to those symptoms. This paper describes the process in generating the corpus and why the documents are relevant to the intended use of the application. The paper then goes on to explain how the application was implemented and the algorithm used to rank the results in the most relevant order to the user's needs. The results from 3 users is tallied and a conclusion on the results drawn.

The Dataset

The corpus used compromises of 267 documents. These documents were scraped from links on Wikipedia pages relating to medical documents. These documents were scraped using a chrome extension called web page downloader. This extension allowed us to download pages specific to medical illnesses. The data that is downloaded is only html documents as this holds the most relevant data for our small corpus. Naturally the magnitude of the domain of medical illnesses and diseases meant the documents we used cover a small subset of every illnesses and diseases and so the queries we used to gather documents were aimed at identifying more common illnesses and ailments. Examples of queries used in building the corpus was "common colds", "prevalent diseases" and "recent illnesses". Additionally, as we initially intended for the

application to gather information specific to certain body parts we catered queries to different parts of the body. An example of this document search was “chest infections”.
The queries used in testing our search engine application were; “Dry cough”, “Pain in chest”, “swollen feet”, “night sweats” and “severe pain”.

The System

The intended use of the system is to provide answers for information needs relating to medical diagnostics or other small ailments. The idea behind this is that, due to the increasing prevalence of both feature phones with internet capability, and widespread 2G/3G network access, realistically, there are more people with internet access than there are with access to doctors or other medical practitioners. Sometimes access to simple diagnostic tools might allow users to self-diagnose, or at the very least glean a better understanding of the possible ailments that they might be under the effects of, with simple solutions for easing symptoms. This system isn’t intended to replace a diagnostician in any way shape or form, but intends to address developmental issues related to limited access to medical resources. We the designers of the system envision that this kind of information retrieval application would be of immense use for those living in rural areas where cell signal penetration might be prevalent but medical access might not. The implementation of our medical information retrieval system is done in Apache Solr. Solr is an open source search platform built on Apache’s Lucene, which is a Java library. The Solr package comes with a number of built in tools, and each of the ones utilised in our system is listed below.

Installation

The Solr package comes in a ZIP, with the bin directory containing the “solr” and “post” command modules, that represent the full functionality of the package.

Solr Admin UI

Once Solr cores are deployed, a user interface is available for the application programmer at localhost:[port]/solr, and is an interface that can be used to manipulate and adjust individual collection parameters in order to tweak performance.

SimplePostTool

Part of Solr’s implementation tools is the SimplePostTool, nested in the example folder of the zipped package is a JAR executable that takes arguments for uploading files of varying formats into collections to be queried through Solr’s RESTful HTTP API.

Solr Browse

Solr Browse is a tool for returning XML formatted search results. The Browse API was used for processing queries and fetching responses, which were then manipulated on the front-end to allow ease of interaction with the search results by the user.

Synonyms (Thesaurus Implementation)

In order to improve the results returned to the user at query time, a thesaurus was used to improve the retrieval of results and improve similarity ranking provided using the Okapi-BM25 ranking algorithm. The thesaurus used was a plaintext collection of terms and equivalent meanings.

BM25

BM25 (Best Matching) is a ranking algorithm that we used in Solr to rank the results in the most relevant order according to a given query. The algorithm ranks a set of documents based on the occurrence of the keywords irrelevant of the relationship and proximity between query terms.

$$BM25=\sum_{i=1}^{W}\frac{TF(i)(1+k)}{TF(i)+k(1-b+b\frac{DL}{avgDL})}IDF(i)$$

$$IDF(i)=\frac{\log(\frac{N-n+1}{n})}{\log(N)}$$

The algorithm we used optimises ranking values by using a b value of 0.75 and k value of 1.2.

Query re-ranking

In addition to the BM25 ranking of results for the initial returned documents, additional query re-ranking was done using the built in re-ranking mechanism built into Solr, that is expanded upon below:

For each of our 5 queries, our initial system implementation took the queries, broke them down into individual terms and did matching for term frequency using BM25. The layout of those results is indicated in the figures below. The format of the queries as passed to Solr's API is as follows:

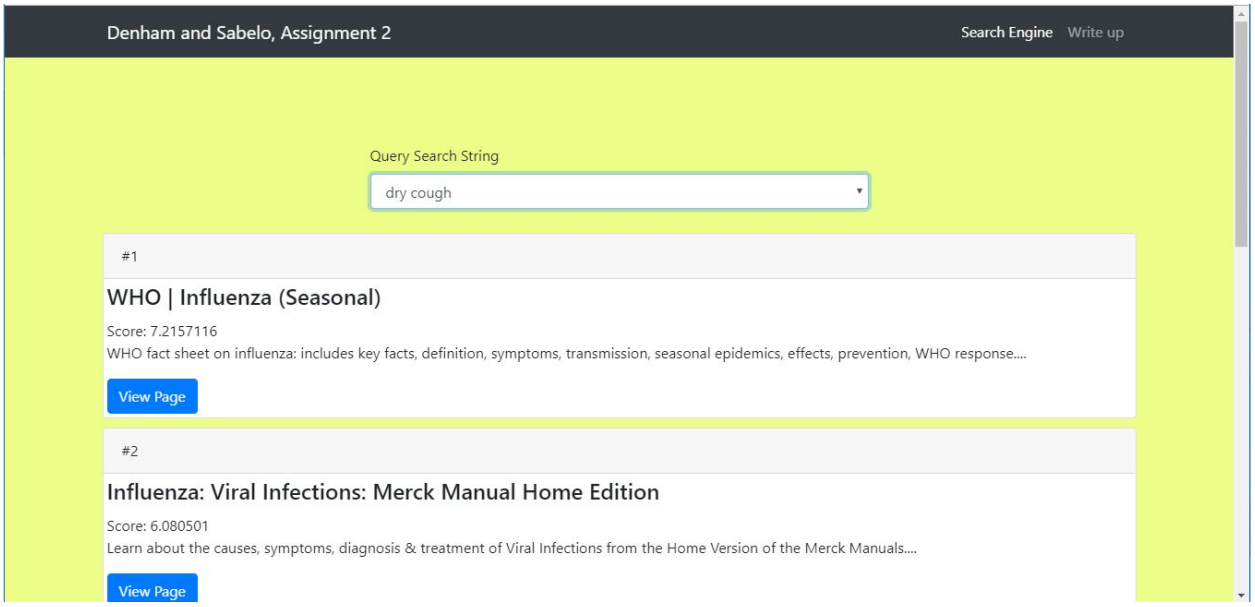
'http:// [url of host]/solr/[collection being queried]/browse?q='query';

For the more intuitive querying setup where users are asked to determine the point of sickness (as a body part) and then indicate severity of pain, as well as the ailment that they feel, a tiered query re-ranking was done. How this works is as follows. The initial n relevant documents

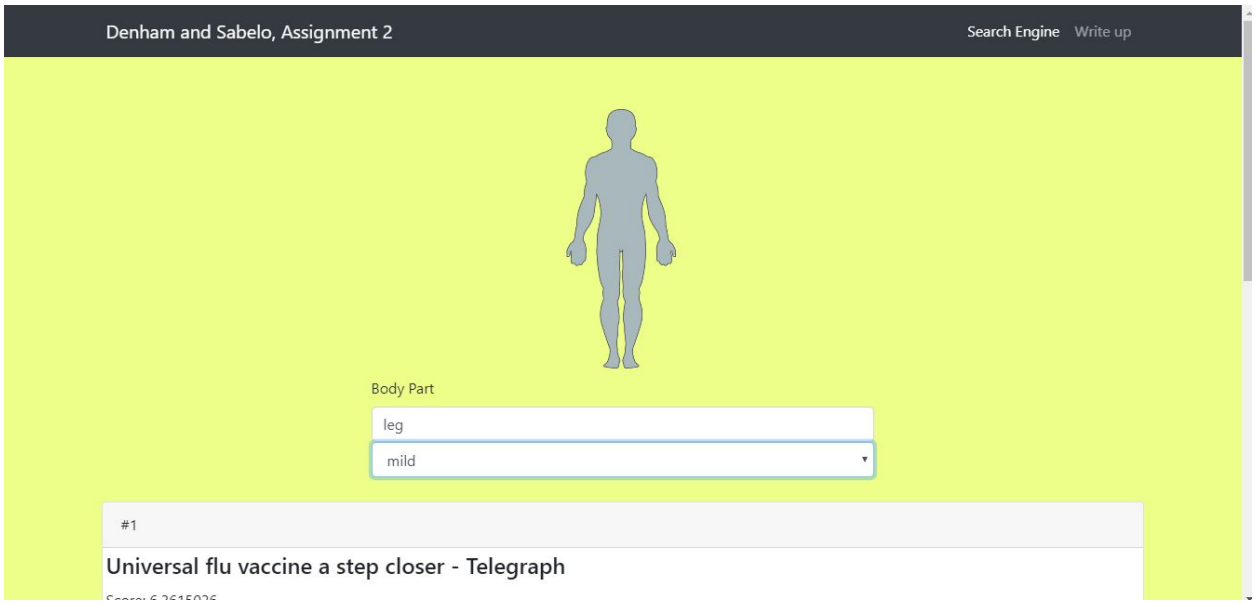
returned by the search are collected based on the initial two portions of the query, those being the body part and severity of pain. These collected documents are then reranked using the additional supplied query (i.e. dry cough) and documents that contain the additional query have their similarity scores increased by a multiplicative factor that is defined in the query. The format of the re-ranking query is as follows:

http:// [url of host]/solr/[collection being queried]/browse?q=[initial query]&rq={!rerank reRankQuery=$rqq reRankDocs=1000 reRankWeight=3}&rqq=[subquery]

This guided query system extended the intended use of the application to select a part of the body and a level of severity/ type of pain on that defined body part, the application returns medical data and illnesses about that body part. This is demonstrated in the mock version in



indexMock2.html (Figure 2 below).



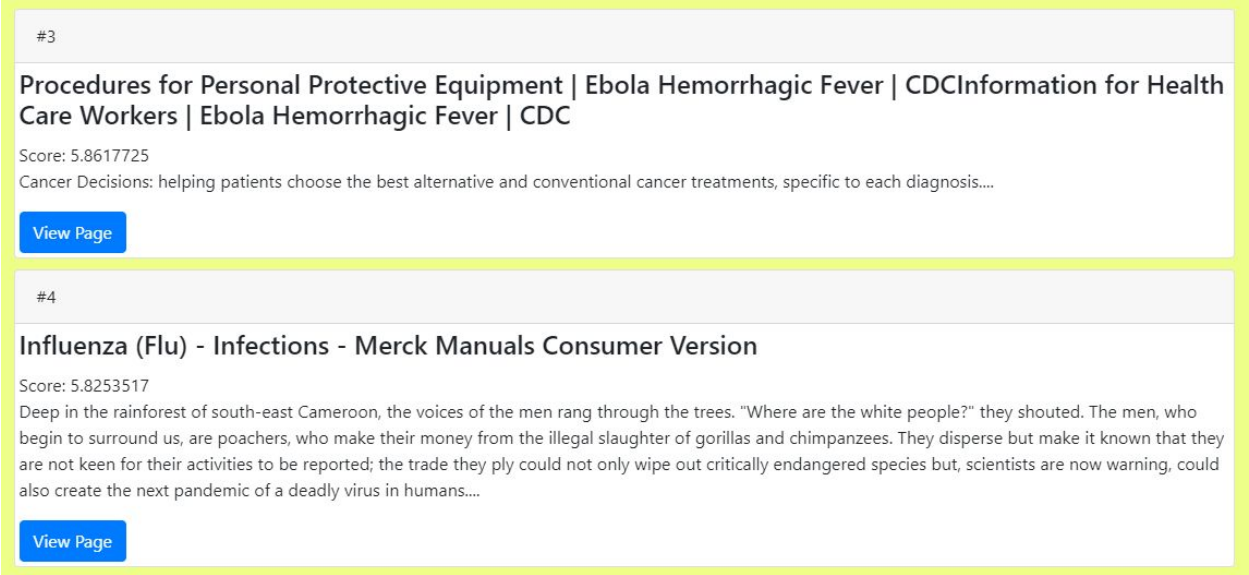
Results

In the controlled test we were assisted by 3 users, Alice, Holly and Jordyn. They were given a brief context of what the purpose of the application was and were asked to select each individual query and comment on whether they felt the results were firstly ordered in a manner that was appropriate and secondly if the results were relevant to the search term. Thus we were able to identify whether users thought the application was ranking relevant documents and separately if the actual results were relevant. The following table illustrates the order that the users felt better represented the query. Moreover the cell is highlighted orange if the user felt the data did not best represent the query and red if it was completely inaccurate to the query.

	Alice	Holly	Jordyn
“Dry cough”	1,2,4,3,5,6	1,2,4,3,5,6	1,2,4,3,5,6
“Pain in chest”	1,2,3,4,5,7,9,6,8	1,2,3,4,5,7,9,8,6	1,2,3,4,5,7,9,8,6
“Swollen feet”	1,2,3,4,5,6	1,2,3,4,5,6	1,2,3,4,5,6
“Night Sweats”	1,2,3,4,5,6	1,2,3,4,5,6	1,2,3,4,6,5
“Severe Pain”	1,2,3,4,5,6,7,8	1,2,3,4,5,6,7,8	1,2,3,4,5,7,8,6

The results indicate the search engine is mostly accurate and furthermore in places where there are inconsistencies, for example on the query “dry cough” everyone agreed 3 and 4 should be

swapped around the scores are 5.8617725 and 5.8253517 for document 3 and 4 respectively. This is a relatively small difference in score and thus we believe to be an acceptable margin of error in the algorithm.



Conclusion

In conclusion, using Apache's Solr search framework as well as built-in tools, we built an information retrieval solution to assist in the early diagnosis and investigation of ailments and illnesses using the considerations and algorithms provided to us throughout the Information Retrieval module, and this project represents an tangible synthesis of our learnings, as a rudimentary version of a solution we think could be vitally effective in a number of contexts. Our testing results indicate that our search platform provides results that users find to be relevant, sorted in a manner that allows for ease of access of resources with a high level of relevant returned search results.