PROJECT REPORT: FIRST RENDITION

Introduction:

The goal of this project is to calculate and return a set of five relevant diseases to the user based on the symptoms that they include as a comma-separated query. The program shall process all symptoms and compare these symptoms to our data set, which is a bunch of diseases and a related list of symptoms. Using one of the various models, the program will return a list of five diseases that it deems relevant. We can check the results via numerous evaluation metrics to see how close our models are compared to each other.

We want to tell the user what ailment they have, but without actual example cases, we cannot test if the ground truth that we arrive at is the correct one. Therefore, our aim is to make this program return ailments as close as we possibly can to the truth. Using a training and test data set, we can avoid overfitting.

Evaluation Paradigm:

The corpus will be a list of all documents with an ID of the name of the disease and a comma-delimited list of symptoms. Queries will search this corpus to find the disease documents that have the most matching symptoms. The test and training data sets will be subsets of the corpus.

The data set that is used for this project can be found at this link: http://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB/index.html

We have no affiliation with the authors of this data set. Likewise, the authors of this data set are not related to our project in any way. As of right now, the corpus is in the incorrect format. However, we were able to create a data set with the correct format, albeit it did take a long time to do so. We will keep on cleaning the corpus so that all of the diseases match the format of the data set. We will also see if we can implement the Count of Disease Occurrence in any way in our program.

Queries will be entered by the user as a comma-delimited list of symptoms. The corpus will then be searched for documents that have the most similar symptoms as the query. The user will get the top 5 results.

The ground truth is what disease the user is actually suffering from. To make this easier to calculate, we are going to only use diseases that are relatively easy to “prove” a person is

suffering from a normal doctor's office (e.g bacterial infections, broken bones, severe conditions such as stroke, heart attack). The disease that matches what the user has is the most relevant.

In theory, we would have data from many doctors who treated patients, recorded their symptoms, and tested them for a given disease. They would use these results to determine the relevance of different diseases compared to a list of symptoms. They may also use the process of elimination to determine the rank of lower-ranking documents. When a doctor examines a person, they may assume a few different illnesses before testing and being certain of what they actually have. The similar illnesses can be seen as relevant, but less relevant than the actual disease. We can increase relevance by number of matching symptoms.

At the moment, we will be using WebMD as our relevance data that we will test against. WebMD is a more advanced website with the same function as ours, so it is a great benchmark for determining what ailment is relevant and what is not relevant. Of course, it is possible that the ailment that we find for our program is not classified as an ailment in the WebMD server; we will tackle that problem when it arises. We will also explore the possibility of creating our own relevance data in the future to avoid discrepancies of ailments involving WebMD.

Methods Evaluated:

The only method that is being compared is the BM25 similarity. In the future, we will be implementing TF-IDF, language models, naive bayes, BIM, and other methods in order to find different ways to return a set of diseases to the user given a query.

Evaluation Methods:

We will be using Precision @ R, MAP, NDCG and Spearman's Coefficient for our current evaluation methods, although not all will be used in this version of the project. Precision @ R will take the amount of diseases returned by WebMD and compare that amount of ailments returned by our program. MAP will take the mean average precision across multiple queries of symptoms, permitting us to determine how accurate our program is for different kinds of diseases. NDGC will allow us to reach a perfect result of 1.0. Finally, even though it isn't implemented yet in our code, Spearman's Coefficient will let us be able to compare two different scoring methods. Since we only have implemented BM25 at this moment, there is no need to implement this evaluation metric at this time.

Results and Typical Errors

The code to run Lucene on a file and the code to calculate the three evaluation methods works at this time. Screenshots of the results of the code will be provided following this section

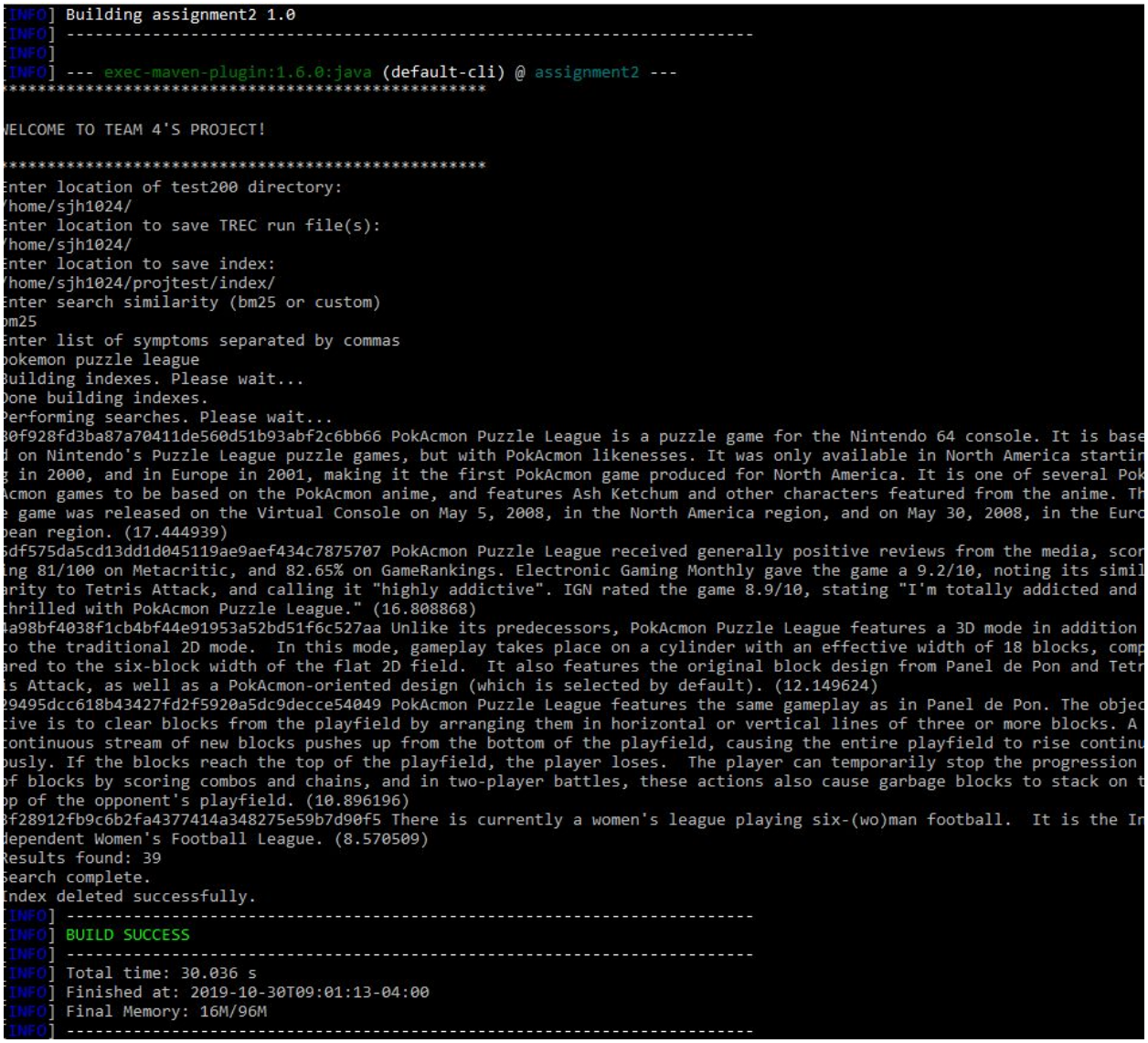
of the report. Any changes we need to make to the evaluation metrics or the indexing will be done before the final rendition of this project is due.

Our problem lies in our data set that is being used for this project. It took a long time for a data set with all of the information needed to be found, as well as a long time to clean the data set and put it in the right format. We had to get from the format in the corpus file to the format in the short data set. This took an estimated 1.5 hours for 15 ailments. Our program can not take the data in its current format and run Lucene on it.

We have two options for how to solve this dilemma. The first option is explore Lucene and find a way for it to take in different file formats that is not cbor. This will allow the program to take in multiple file formats, and would add a neat additional feature to the program. The second option is to change the format of the corpus and data set to a file format that Lucene can take in. This is not optimal, but it may be a choice we have to make if we run low on time.

See the next page for the beginning of the relevant screenshots.

Relevant Screenshots of Results



This is our output for our program on a regular cbor file. This is a slightly modified version of Assignment 1, and it successfully returns 5 results from a query. If we can get our data set to work with our program, then we will just need to change the input file in order to switch our output to the correct one.

```
## [1,] "For query: " "179"
##  [1] 0.01694915 0.03389831 0.05084746 0.06779661 0.08474576
0.10169492
##  [7] 0.11864407 0.13559322 0.15254237 0.16949153 0.18644068
##                    i
## [1,] "For query: " "180"
##  [1] 0.03225806 0.06451613 0.09677419 0.12903226 0.16129032
0.19354839
##  [7] 0.22580645 0.25806452 0.29032258 0.32258065 0.35483871
0.38709677
## [13] 0.41935484
##                    i
## [1,] "For query: " "181"
## [1] 0.09090909 0.18181818
##                    i
## [1,] "For query: " "182"
## [1] 0.03571429 0.07142857
##                    i
```

Here is some example output for the Precision at R part of the R code. We'll be looking to optimize the output format in the future to make the results clearer.

```
nDcg(qIdVector,docList,sortedDocList)

## [1] 0.02570835
```

Finally, here is the output for the NDCG part of the R code. This is a much more direct output, as there is only one value to show. MAP will also return one value, which is 0.3724176, although it was not outputted in the files provided.

Contributions and Roles

Ben has the role of cleaning our datasets and providing mathematical insight. He will be implementing language models with smoothing and will be implementing BIM as an alternative to BM25 for our system. Sarah has the role of formatting/organizing written documents and properly styling code so it is easier for everyone to read. She will be implementing stemming for query tokens so our system understands user input, a vector space model for the corpus in which each disease is a document represented by a vector with diseases as its terms, and Spearman's

Rank Correlation Coefficient to evaluate our system versus the ground truth. Talha has the role of creating the training and test sets of our data to utilize, and evaluating the results of the project. He will be using MAP, NDCG, and Precision@R to evaluate our system's performance, and he will be using Naive Bayes to train our system.