Search Engine Report

UCSC CSE272 Information Retrieval Hw1

Omkar Ghanekar

oghaneka@ucsc.edu, SID: 1926974

1 Problem Statement

Create a search engine for the given medical documents and evaluate the performance on trec_eval for different ranking algorithms.

The code for the search engine can be found here.

2 Software

I used ElasticSearch for creating the search engine. Elastic Search is a distributed software built on top of Apache Lucene. It provides REST APIs and JSON based schemas for documents. It required a 2 step setup process. The ElasticSearch (ES) server and a python based ES client. The overall architecture of the system is described in the figure below.

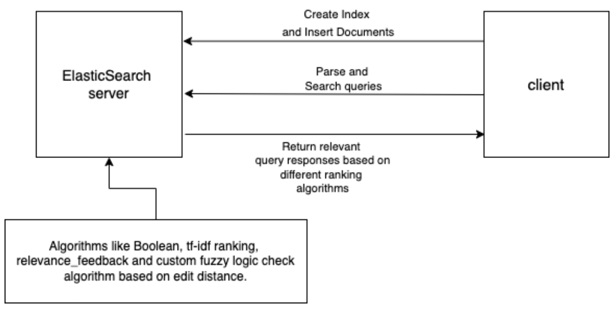

ElasticSearch server
client
Create Index and Insert Documents
Parse and Search queries
Return relevant query responses based on different ranking algorithms
Algorithms like Boolean, tf-idf ranking, relevance_feedback and custom fuzzy logic check algorithm based on edit distance.

Figure 1: System Architecture

The major advantage provided by ElasticSearch is the HTTP based API calls to insert documents into and query an index. But this comes at a tradeoff for the overall control on the algorithm provided by Lucene which is not there in ElasticSearch for some functions.

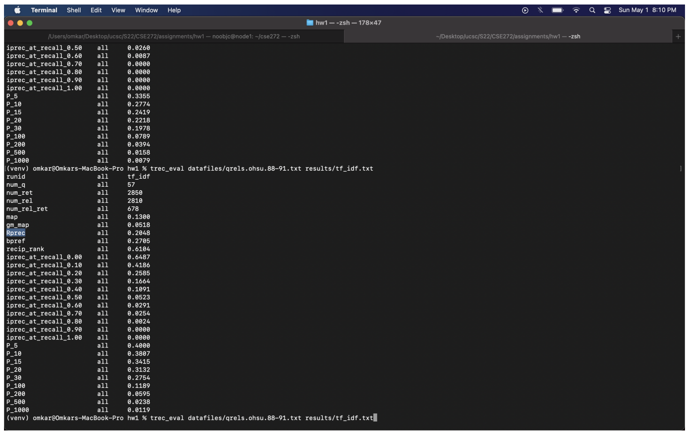
3 Algorithms

I implemented the boolean query, tf and tf_idf ranking algorithms for querying the index based on the query files. Analyzers and tokenizers were added when creating the index to create an inverted list of indexes for faster retrieval. Stemming is also done to make words like (eating, eats, eaten) point to same base word (eat). The MedID is the primary key or the id of the index. (**Trec dataset readme**) Documents are added into the index with a *settings* and a *mapping* fields which help in defining the fields, their filters and write custom analyzers or tokenizers for the same. Query handling is done in a similar way on ElasticSearch, the queries and tokenized and stemmed and then mapped to the documents in the inverted index list. ElasticSearch provides a *bool* query and a search query API which takes relevant query and its fields as an input and returns a set of documents matching them based on the score(relevance) of the document.

The Relevance feedback algorithm was a wonderful learning curve for me, both from understanding of the algorithm and from the coding perspective. The relevance file from the 87 set is used for inducing the feedback feature into the search results. The first step was to parse the relavant file and query file to extract the important features and pass them on to the search function. The search function takes in an additional parameter stating the ratings of the document with given ID.

3.1 Custom Ranking Algorithm

I used fuzzy logic to implement an edit distance algorithm which checks the tokens in the query and returns documents which match the query with an edit distance of 0,1 or 2. After some testing, the results were surprisingly improved. For an edit distance of 0 the trec_eval results were pretty similar to the tf_idf ranking. For the fuzzy value of 2, the scores(relevance) was much lower due to high distortion in the original query search. But for an edit distance(fuzzy value) of 1, the search boosted the trec_eval results to a Rprec of 0.4 from the 0.2 of tf_idf.



Terminal
Shell
Edit
View
Window
Help
hw1 — -zsh — 178×47
/Users/omkar/Desktop/ucsc/S22/CSE272/assignments/hw1 — noobjc@node1: ~/cse272 — -zsh
~/Desktop/ucsc/S22/CSE272/assignments/hw1 — -zsh

```
iprec_at_recall_0.50    all     0.0260
iprec_at_recall_0.60    all     0.0087
iprec_at_recall_0.70    all     0.0000
iprec_at_recall_0.80    all     0.0000
iprec_at_recall_0.90    all     0.0000
iprec_at_recall_1.00    all     0.0000
P_5                     all     0.3355
P_10                    all     0.2774
P_15                    all     0.2419
P_20                    all     0.2218
P_30                    all     0.1978
P_100                   all     0.0789
P_200                   all     0.0394
P_500                   all     0.0158
P_1000                  all     0.0079
(venv) omkar@Omkars-MacBook-Pro hw1 % trec_eval datafiles/qrels.ohsu.88-91.txt results/tf_idf.txt
runid                   all     tf_idf
num_q                   all     57
num_ret                 all     2850
num_rel                 all     2810
num_rel_ret             all     678
map                     all     0.1300
gm_map                  all     0.0518
Rprec                   all     0.2048
bpref                   all     0.2705
recip_rank              all     0.6104
iprec_at_recall_0.00    all     0.6487
iprec_at_recall_0.10    all     0.4186
iprec_at_recall_0.20    all     0.2585
iprec_at_recall_0.30    all     0.1664
iprec_at_recall_0.40    all     0.1091
iprec_at_recall_0.50    all     0.0523
iprec_at_recall_0.60    all     0.0291
iprec_at_recall_0.70    all     0.0254
iprec_at_recall_0.80    all     0.0024
iprec_at_recall_0.90    all     0.0000
iprec_at_recall_1.00    all     0.0000
P_5                     all     0.4000
P_10                    all     0.3807
P_15                    all     0.3415
P_20                    all     0.3132
P_30                    all     0.2754
P_100                   all     0.1189
P_200                   all     0.0595
P_500                   all     0.0238
P_1000                  all     0.0119
(venv) omkar@Omkars-MacBook-Pro hw1 % trec_eval datafiles/qrels.ohsu.88-91.txt results/tf_idf.txt
```

Figure 2: Results for the search query

4 Learnings and Outcome

This homework provided me a wonderful introduction to the world of IR systems and tremendous power of a search engine. I learned ElasticSearch and data preprocessing for the queries and documents. The assignment was worth the time spent working on it. This will also help me in my course project moving forward which is based on a search engine for Wikipedia.