# FINDING DOCTOR: A search engine

Submitted by:

**Sabarnath Maddinieni(scm345)**

**Hari Bugide(hkb44)**

**Shyam prasad Sukumar(ss4978)**

**Chinduri Deivasagayam(cd3266)**

## Introduction:

Our project goal is to develop a search engine that retrieves relevant information about doctors from a set of static documents. We used the information of doctors located in the east coast states of the USA.

## Purpose:

The purpose of the search engine is to provide a simple interface for the user to retrieve the relevant information based on the user inputs. This system requires Doctor Specialty, City, State as user inputs to retrieve the relevant doctor's information. By using this search engine, the system would display the most relevant search results at the top. For example, if the user needs information about cardiologists in a certain city of a state, he will be provided with relevant doctors' information by avoiding other irrelevant results.

## Data:

Our search engine is related to the medical domain and it retrieves information about the doctors available in a particular city of the state.

We have downloaded the dataset from healthdata.gov, which was a huge dataset, consisted of detailed information of doctors available across the USA. Our project mainly focuses on the doctors who are available in the east coast region of the USA, hence we have filtered major states from our dataset. We included information from CT(Connecticut), DE(Delaware), FL(Florida),

GA(Georgia), ME(Maine), MA(Massachusetts), NC(North Carolina), NH(New Hampshire), NJ(New Jersey) states to build our search engine.

We had to remove unwanted, repeated, and empty fields that were not relevant to our search results. “Zip code” column in our dataset had both Zip code and Ext code. For ease of our end-users, we have truncated the Ext code and displayed only the first 5 digits of the Zip code.

## **Use Cases**:

The basic information a user receives, as a result, would be Doctor’s Name, Gender, Address, Contact details, hospital name, organization name, and graduation year.

- Public:

  The residents are the primary users of our engine. The users can know the details of available doctors with a required specialty they are looking for in a particular city of a state.
- Medical Representatives:

  The search engine also can be used by Medical Representatives who require details of doctors or hospitals to sell their medicines.

- Business:

  People who would like to develop their mode of business would target the places where people’s basic facilities are available. Hence by getting the details of doctors/hospitals available in a city they can start their business. Also, by knowing the existing doctors with a primary skill, the doctors/hospitals with other specialties can be established in the same locality.

## **SYSTEM FEATURES:**

To achieve our goal, we have considered certain fields to be mandatory for the search, so the user needs to provide those fields information to retrieve the relevant documents.

Information required to perform the search:

- Primary specialty
- City
- State

We have created an index on the elastic cluster server to host our preprocessed data, which we are retrieving using our search engine. We have declared the mandatory field's type as “text” and applied a standard analyzer. We mapped city and state field to the Boolean similarity as these fields content are short, and the user primary goal is to get information based on those fields. We mapped primary specialty to BM_25 similarity so that the retrieval is performed based on the user inputs appearing in each document regardless of their proximity within the document. For example, if a user enters ‘cardiology’ for Primary Specialty doctors, information related to cardiology is retrieved first.

Our search query should contain 3 fields and should follow the below structure:

Primary Specialty, City, State.

We will be using the above three fields for the potential matches to retrieve the data.

We will be using the above three fields for scoring (ranking). We are using Boolean mapping for state and city as full-text ranking is not needed and the score can be based on wheatear the user input is present or not, and the BM_25 similarity technique for Primary specialty, so that document that has the best match of the query term in the primary specialty field is retrieved first.

We are not performing boosting on any of the fields because we have made Primary Specialty, City, State fields mandatory for the search, boosting them will not have much effect. Boosting can be applied for Primary Specialty, but this is not effective in our case.

We have created the index as “hkb44_info624_201904_final_doc_data” and mapped mandatory fields by using the below code.

## Code snippets for creating an index with custom similarities.

```
PUT / hkb44_info624_201904_final_doc_data
{
  "settings": {
    "index": {
      "similarity": {
"my_bm25": { "type": "BM25", "k1": 2.0, "b":1.0
                    },

"my_dfr": {"type": "DFR", "basic_model": "g", "after_effect": "l", "normalization": "h2",
"normalization.h2.c": "3.0"}
                  }
             }
              }
}
```

For uploading the data to the above-created index, we have used the Kibana interface.

## Code snippets for mapping the mandatory fields to use custom similarities for the retrieval.

**Mapping Primary specialty:**

```
    PUT / hkb44_info624_201904_final_doc_data/_mapping
{
  "properties" : {
"Primary specialty" : {
"type" : "text", "analyzer": "english", "similarity" : "my_bm25"
            } }
}
```

**Mapping City:**

PUT / hkb44_info624_201904_final_doc_data/_mapping

{

"properties" : {

"City" : {

"type" : "text", "analyzer": "english", "similarity" : "boolean"

} }

}

**Mapping State:**

PUT / hkb44_info624_201904_final_doc_data/_mapping

{

"properties" : {

"State" : {

"type" : "text", "analyzer": "english", "similarity" : "boolean"

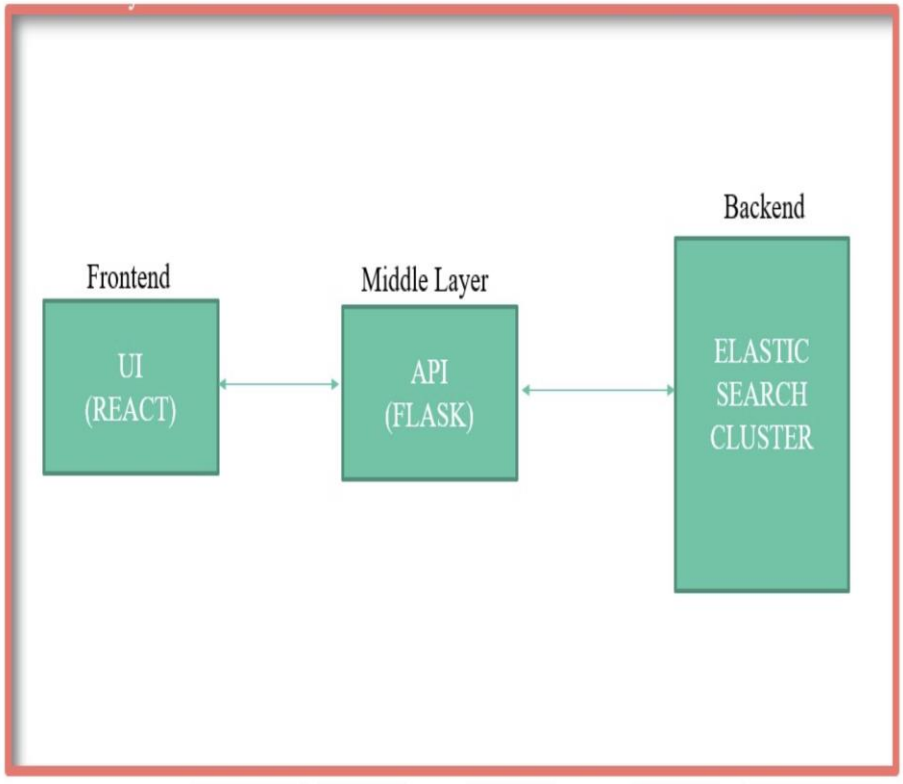
} }

}

## Code snippets of retrieval query :

```
GET /hkb44_info624_201904_final_doc_data/_search
{
"from" : 0, "size" : 10,
  "query": {"bool": {"should": [
    {"match": {
      "State":"cardiology newark nj"
   }},
    {"match": {
      "City": "cardiology newark nj"
   }},
   {"match_phrase_prefix": {
      "Primary_specialty": "cardiology"
   }}]}
}
}
```

## System Architecture:



**Search Evaluation:**

**First Use case:**

Search query: Cardiology in Newark NJ

Information needed: Cardiology doctors from Newark city of NJ state

Documents Retrieved:10

Documents Relevant:4

Documents non-Relevant:6

Total Documents: 8667

| | Relevant | Nonrelevant |
|---|---|---|
| Retrieved | 4(TP) | 6(FP) |
| Not Retrieved | 0(FN) | 8657(TN) |

$$\boldsymbol{Precision(P)} = \frac{TP}{(TP + FP)} = \frac{4}{10} = 0.4$$

$$\boldsymbol{Recall(R)} = \frac{TP}{(TP + FN)} = \frac{4}{4} = 1$$

$$\boldsymbol{F1\ score} = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} = 2 * \frac{(0.4)}{(1.4)} = 0.571$$

$$\boldsymbol{DCG_5} = rel_1 + \frac{rel_2}{log_2\,(2)} + \frac{rel_3}{log_2\,(3)} + \frac{rel_4}{log_2(4)} + \frac{rel_5}{log_2(5)}$$

$$\boldsymbol{DCG_5} = 1 + \frac{1}{1} + \frac{1}{1.584} + \frac{1}{2} + 0$$

$$\boldsymbol{DCG_5} = 1 + 1 + 0.631 + 0.5$$

$$\boldsymbol{DCG_5} = 3.13$$

Here we are getting our result in order from relevant to non-relevant and considered relevance value as 1 and non-relevant value as 0,hence IDCG will also be same as DCG.

$$\boldsymbol{nDCG} = \frac{DCG}{IDCG}$$

$$\boldsymbol{nDCG} = \frac{3.13}{3.13} = 1$$

**Second Use case:**

Search query: oncology, Augusta, ga

Information needed: oncology from Augusta city of GA state

Documents Retrieved:10

Documents Relevant:6

Documents non-Relevant:4

Total Documents: 8667

| | Relevant | Nonrelevant |
|---|---|---|
| Retrieved | 6(TP) | 4(FP) |
| Not Retrieved | 0(FN) | 8657(TN) |

$$\boldsymbol{Precision(P)} = \frac{TP}{(TP + FP)} = \frac{6}{10} = 0.6$$

$$\boldsymbol{Recall(R)} = \frac{TP}{(TP + FN)} = \frac{6}{6} = 1$$

$$\boldsymbol{F1\ score} = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} = 2 * \frac{(0.6)}{(1.6)} = 0.75$$

$$\boldsymbol{DCG_5} = rel_1 + \frac{rel_2}{log_2\,(2)} + \frac{rel_3}{log_2\,(3)} + \frac{rel_4}{log_2(4)} + \frac{rel_5}{log_2(5)}$$

$$\boldsymbol{DCG_5} = 1 + \frac{1}{1} + \frac{1}{1.584} + \frac{1}{2} + \frac{1}{2.321}$$

$$\boldsymbol{DCG_5} = 1 + 1 + 0.631 + 0.5 + 0.430$$

$$\boldsymbol{DCG_5} = 3.56$$

Here we are getting our result in order from relevant to non-relevant and considered relevance value as 1 and non-relevant value as 0,hence IDCG will also be same as DCG.

$$\boldsymbol{nDCG} = \frac{DCG}{IDCG}$$

$$\boldsymbol{nDCG} = \frac{3.13}{3.13} = 1$$

**Third Use case:**

Search query: General Surgeon, Miami, FL

Information needed: General surgeon from Miami city of FL state

Documents Retrieved:10

Documents Relevant:8

Documents non-Relevant:2

Total Documents: 8667

| | Relevant | Nonrelevant |
|---|---|---|
| Retrieved | 8(TP) | 2(FP) |
| Not Retrieved | 0(FN) | 8657(TN) |

$$\boldsymbol{Precision(P)} = \frac{TP}{(TP + FP)} = \frac{8}{10} = 0.8$$

$$\boldsymbol{Recall(R)} = \frac{TP}{(TP + FN)} = \frac{8}{8} = 1$$

$$\boldsymbol{F1\ score} = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} = 2 * \frac{(0.8)}{(1.8)} = 0.889$$

$$\boldsymbol{DCG_5} = rel_1 + \frac{rel_2}{log_2\,(2)} + \frac{rel_3}{log_2\,(3)} + \frac{rel_4}{log_2(4)} + \frac{rel_5}{log_2(5)}$$

$$\boldsymbol{DCG_5} = 1 + \frac{1}{1} + \frac{1}{1.584} + \frac{1}{2} + \frac{1}{2.321}$$

$$\boldsymbol{DCG_5} = 1 + 1 \; + 0.631 + 0.5 + 0.430$$

$$\boldsymbol{DCG_5} = 3.56$$

Here we are getting our result in order from relevant to non-relevant and considered relevance value as 1 and non-relevant value as 0,hence IDCG will also be same as DCG.

$$\boldsymbol{nDCG} = \frac{DCG}{IDCG}$$

$$\boldsymbol{nDCG} = \frac{3.13}{3.13} = 1$$

When we observe the results, we can infer that our search engine was able to retrieve all the relevant documents at the top(with in 10documents)keeping False-negative to zero i.e. all the document which satisfy our information needs are retrieved.

When we are tried to change the custom similarity technique to my_dfr we noticed that the information retrieved both the cases tends to be the same.

Our search engine index is located at one of the following elastic search nodes tux-es1.cci.drexel.edu:9200,tux-es2.cci.drexel.edu:9200,tux-es3.cci.drexel.edu:9200 with the below index name:

**hkb44_info624_201904_final_doc_data.**

**Experiences**:

While working on developing this system implementation we were able to explore technologies which involve in information retrieval. We also gained knowledge on how to store and organize our data to perform the retrieval task. We faced difficulties in passing the user inputs to our retrieval body, but we were able to handle the issue using flask(API).

**Limitations**:

A major limitation for our search engine is we restricted our data to only certain states of the USA and we require Primary specialty, City, and state information to perform the search if the user fails to provide this information appropriately and search for information of the states which are not included in the database no documents are retrieved.

**Future Work**:

Future work of the project is to include more states' information on the clusters and styling the search engine along with user input validation.

**Conclusion**:

“Finding Doctor” search engine was designed to get the information of the doctors based on their primary specialty and location we can achieve this task if doctor information is present on the clusters.

**Technologies used for system implementation**:

React(For user interface)

Flask( To implement API)

Elastic cluster(To host the document data)

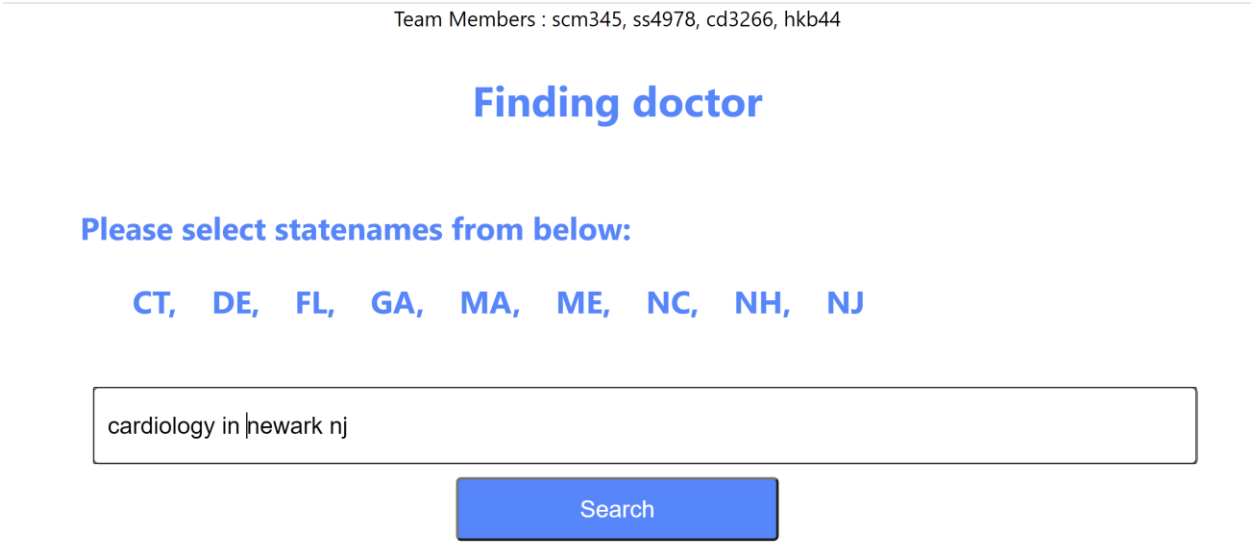
**System requirements to run the Search Engine**:

1. Any operating system which has node js, npm, python installed and connected to 'vpn.drexel.edu' VPN network.

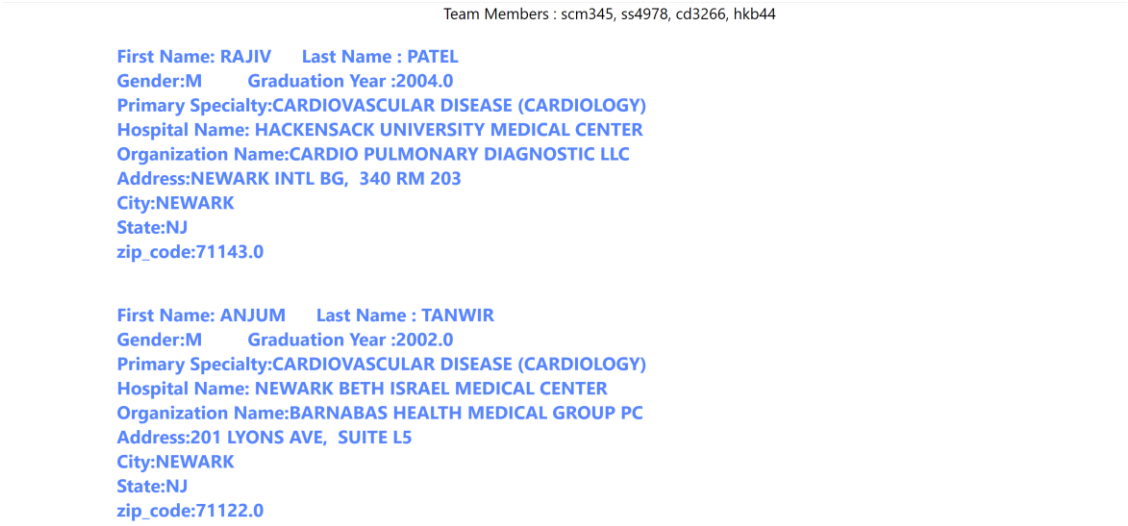
**Code Repository link:**

https://github.com/HariKumarBugide/INFO-624-Final-Project

Search Engine Screenshots:



For valid inputs




For invalid inputs: