INFO 624 – Final Project
News Search Engine
Group 4

Project Type: Search Engine Building

Team Members:

Nupur Roy Chowdhury, nr572@drexel.edu

Mahesh Sercat Ramakumar, ms4976@drexel.edu

Rohith Lakshminarayana, rl669@drexel.edu

Manisha Uttam Nandawadekar, mun24@drexel.edu

Table of Contents

0.1 Report

• Why (Purpose of the Search Engine):

In our day to day life, searching is a social routine. We tend to search on various topics. Depending on the information what we search the data boundary is sometimes smaller or it is vast and tends to get increased if the information searched has a broader perspective.

Hence, the need for a search engine arises against which assists us in finding relative information from the vast volumes of data. The main purpose of building our search engine is to create a dedicated news-based search engine where the users are able to find all the news related data from different and relevant news channels to their search query. Depending on whether the user wants to view the news from a particular news channel too if the user wants to follow only the latest news on a given query.
Finding the relevant information when there is more information available is the most challenging approach to search and our search engine caters to this need of the user.

• What (Data and Domain):

Our search engine index is built on the news domain. Our data consist of the source from where the news has originated, the title of the news, the description of the news, the content which says furthermore about a piece of particular news, when the news has been published, and who is the author of this news and the article appearing in the news channel.

We have used News API (http://newsapi.org/) to collect our data.

Field	**Description**	**Data type**
Source	The identifier id and a display name for the source this article came from.	Object
Author	The author of the article	String
Title	The headline or title of the article.	String
Description	A description or snippet from the article.	String
url	The direct URL to the article.	String

urlToImage	The URL to a relevant image for the article.	String
publishedAt	The date and time that the article was published, in UTC (+000).	String
content	The unformatted content of the article, where available. This is truncated to 200 chars.	String

Underneath represent the steps which we have done to collect our data:

• We are establishing the connection to the Elasticsearch servers hosted in “Drexel Network”.

• Once the connection has got established, we are collecting it in a variable called es. This variable was used to construct the index.

• We have collected a list of query terms to collect our data from the News API.

• Creating an index with default settings and mappings.

•The primary index where our data is accumulated is: **ms4976_info624_201904_newsproject**

• Who (Users of the Search Engine):

As the search engine has been built on news related data, it will serve the users who demand information about day to day news. The users can use the news search engine and search for top headlines/most recent news which servers their basic information needs.

The use cases (information needs) based on which the search engine was built is:

1. Use case related to the title: at this place, the users can search the news search engine based on the title of the news which they have in their mind or want to perceive more information about. Example: ‘crimes in US’ or ‘US elections’ which provides an extremely brief description of the news.

2. Use case related to the source: here users can search based on which source the news originated from i.e. which news channels news they would be interested to read.

3. Use case related to the description: Since the description is more on a broader perspective if the user has any keyword in their mind utilizing what they want to search the search engine they would be able to do that. For example, they have a keyword: COVID the user would be able to search the news search engine using that keyword and discover the related news from various sources/channels.

4. Use case related to the author: there are times when the user wants to follow or refer to a particular piece of news reported by certain authors. Using this news search engine, they would be capable of searching the news based on the author they would like to read.

• How (Process of Building Search Engine):

Before we have started building the search engine, we have made some decisions related to the data which we would collect and feed into our primary index created on the elastic search servers. The decisions obtain are as follows in the **jupyter notebook (Data Collection and indexing):**

a) As we have already created an index and have loaded around 4,221 Documents which were extracted from various news sources, we have performed some data-preprocessing tasks on our collected data.

b) As the data are collected in JSON format when we had a look at the data, we could find that the source field which consists of two columns: name and id had the id column empty in few cases.


```
"articles": [
    {
        "source": {
            "id": null,
            "name": "Lifehacker.com"
        },
        "author": "Elizabeth Yuko",
```


c) Hence, here we have removed the id and considered only the name field inside the source field.

d) We have removed content and urlToImage fields from the data.

e) Using the field 'publishedAt' we have generated a field called timestamp. We have normalized this value and have added the rank_feature to this field. So, that, when we explore for the data the most recent data, will appear first and followed subsequently.

f) Next, we are removing redundant data. Where we are indexing unique documents and have been uncollected previously. Based on this, the documents will be indexed from the last index previously created.

g) Total 3561 documents indexed successfully. After indexing, Total 3561 documents are present in the index: **ms4976_info624_201904_newsproject**

1. The data field related to the project along with the analyzer which we have applied for each field is as below:

```
index_name = 'ms4976_info624_201904_newsproject'

request_body = {
        'mappings': {

            "properties":{
                "source":{
                    "type": "text",
                    "analyzer": "standard"
                    },
                "author":{
                    "type": "text" ,
                    "analyzer": "standard",
                    "similarity": "boolean"
                    },
                "title":{
                    "type": "text" ,
                    "analyzer": "english",
                    },

                "description":{
                    "type": "text" ,
                    "analyzer": "english",
                    },
                 "url":{
                    "type": "text"
                    },

                "publishedAt":{
                    "type" : "date"
                    },
                "timestamp" :{
                    "type" : "rank_feature",
                    "positive_score_impact" : True
                    }

                }
```

S. No	Field	Type	Analyzer	Similarity
1	Source	Text	Standard	
2	Author	Text	Standard	Boolean
3	title	Text	English	
4	description	Text	English	
5	url	text		
6	publishedAt	date		
7	timestamp	Rank_feature		

2. The comprehensive description of the search queries based on the specific use cases:

1.Use case related to the title:

Use case 1: "COVID-19" on index "ms4976_info624_201904_newsproject"
Query Keyword: "COVID-19"

```
es.search(index = 'ms4976_info624_201904_newsproject', body=
{"from":0, "size":10,
 "query":
 {
      "bool":
     {
       "must":
      [{
       "multi_match":
      {
                "query": "COVID-19",
                "fields": ["title","description"]
       }
      }],
       "should":
      {
       "rank_feature":
      {
      "field": "timestamp",
        "sigmoid":{"pivot": 5,"exponent":0.6}
      }
      }
      }
       }
     })
```

- The field we are searching here for the potential matches is title and description.
- The ranking/scoring feature has been uniquely defined for the field timestamp.

2.Use case related to the source:

Use case 2: Searching articles based on the source.
Query Keyword: " articles by BBC news"

```
es.search(index = 'ms4976_info624_201904_newsproject', body=
{"from":0, "size":10,
 "query":
 {
  "bool":
  {
   "must":
   [{
     "multi_match":
     {
      "query": "articles by BBC news",
      "fields": ["source","title","description"]
     }
   }],
   "should":
   {
    "rank_feature":
    {
     "field": "timestamp",
     "sigmoid":{"pivot": 5,"exponent":0.6}
    }
   }
  }
 }
})
```

- The field we are searching here for the potential matches is source, title and description.

3. Use case related to the description:

Use case 3: Searching articles based on the description.
Query Keyword: " US elections"

```
GET ms4976_info624_201904_newsproject/_search
{
 "from":0, "size":10,
 "query":
 {
  "bool":
  {
   "must":
   [{
     "multi_match":
     {
      "query": " US elections ",
      "fields": ["description"]
     }
   }],
   "should":
   {
    "rank_feature":
    {
     "field": "timestamp",
     "sigmoid":{"pivot": 5,"exponent":0.6}
    }
   }
  }
 }
}
```

- The field we are searching here for the potential matches is description.

1. **Use case related to the author:**

Use case 3: Searching articles based on the author.
Query Keyword: " Annie Karni articles on Trump"

```
GET ms4976_info624_201904_newsproject/_search
{
```

```
"from":0, "size":10,
"query":
{
 "bool":
 {
  "must":
  [{
    "multi_match":
    {
     "query": "Annie Karni articles on Trump",
     "fields": ["author","title","description"]
    }
  }],
  "should":
  {
   "rank_feature":
   {
    "field": "timestamp",
    "sigmoid":{"pivot": 5,"exponent":0.6}
   }
  }
 }
}
```

- The field we are searching here for the potential matches is description

Selected similarity, scoring and boosting methods:

- **Similarity: (Jupyter Notebook used is: Custom Similarites Comparison)**

  For the similarity we have experimented with the below similarities on different indices we have created for the project.

a. index_name = 'ms4976_info624_201904_newsproject1'

```
#creating an below index with respective settings
index_name = 'ms4976_info624_201904_newsproject1'
request_body = {
        "settings":{
            "index":{
                "similarity":{
                    "custom_bm25":{
                        "type": "BM25",
                        "k1": 2.0,
                        "b":1.0
                    },
                    "custom_dfr":{
                        "type": "DFR",
                        "basic_model": "g",
                        "after_effect": "l",
                        "normalization": "h2",
                        "normalization.h2.c": "3.0"
                    }
                }
            }
    },

        'mappings': {

            "properties":{
                "source":{
                    "type": "text",
                    "analyzer": "standard"
                    },
                "author":{
                    "type": "text" ,
                    "analyzer": "standard",
                    "similarity": "boolean"
                    },
                "title":{
                    "type": "text" ,
                    "analyzer": "english",
                    "similarity":"custom_dfr"
                    },

                "description":{
                    "type": "text" ,
                    "analyzer": "english",
                    "similarity":"custom_bm25"
                    },
                 "url":{
                    "type": "text"
                    },

                "publishedAt":{
                    "type" : "date"
                    },
                "timestamp" :{
                    "type" : "rank_feature",
                    "positive_score_impact" : True
                    }
                }
            }
        }
es.indices.create(index = index_name, body = request_body)
```

```
{'acknowledged': True,
 'shards_acknowledged': True,
 'index': 'ms4976_info624_201904_newsproject1'}
```

b. index_name = 'ms4976_info624_201904_newsproject2'

```
#creating an below index with respective settings
index_name = 'ms4976_info624_201904_newsproject2'
request_body = {
        "settings":{
            "index":{
                "similarity":{
                    "custom_bm25":{
                        "type": "BM25",
                        "k1": 1.5,
                        "b":1.0
                    },
                    "custom_dfr":{
                        "type": "DFR",
                        "basic_model": "if",
                        "after_effect": "b",
                        "normalization": "h3",
                        "normalization.h2.c": "3.0"
                    }
                }
            }
    },

        'mappings': {

            "properties":{
                "source":{
                    "type": "text",
                    "analyzer": "standard"
                    },
                "author":{
                    "type": "text" ,
                    "analyzer": "standard",
                    "similarity": "boolean"
                    },
                "title":{
                    "type": "text" ,
                    "analyzer": "english",
                    "similarity":"custom_dfr"
                    },

                "description":{
                    "type": "text" ,
                    "analyzer": "english",
                    "similarity":"custom_bm25"
                    },
                 "url":{
                    "type": "text"
                    },

                "publishedAt":{
                    "type" : "date"
                    },
                "timestamp" :{
                    "type" : "rank_feature",
                    "positive_score_impact" : True
                    }
                }
            }
        }
es.indices.create(index = index_name, body = request_body)
```

```
{'acknowledged': True,
 'shards_acknowledged': True,
 'index': 'ms4976_info624_201904_newsproject2'}
```

c. index_name = 'ms4976_info624_201904_newsproject3'

```
#creating an below index with respective settings
index_name = 'ms4976_info624_201904_newsproject3'
request_body = {
        "settings":{
            "index":{
                "similarity":{
                    "custom_bm25":{
                        "type": "BM25",
                        "k1": 1.0,
                        "b":0.9
                    }
                }
            }
        },

        'mappings': {

            "properties":{
                "source":{
                    "type": "text",
                    "analyzer": "standard"
                    },
                "author":{
                    "type": "text" ,
                    "analyzer": "standard",
                    "similarity": "boolean"
                    },
                "title":{
                    "type": "text" ,
                    "analyzer": "english",
                    "similarity":"custom_bm25"
                    },

                "description":{
                    "type": "text" ,
                    "analyzer": "english",
                    "similarity":"custom_bm25"
                    },
                 "url":{
                    "type": "text"
                    },

                "publishedAt":{
                    "type" : "date"
                    },
                "timestamp" :{
                    "type" : "rank_feature",
                    "positive_score_impact" : True
                    }
                }
            }
        }
es.indices.create(index = index_name, body = request_body)
```

```
{'acknowledged': True,
 'shards_acknowledged': True,
 'index': 'ms4976_info624_201904_newsproject3'}
```

d. index_name = 'ms4976_info624_201904_newsproject4'

```
#creating an below index with respective settings
index_name = 'ms4976_info624_201904_newsproject4'
request_body = {
        "settings":{
            "index":{
                "similarity":{
                    "custom_dfr":{
                        "type": "DFR",
                        "basic_model": "ine",
                        "after_effect": "b",
                        "normalization": "z",
                        "normalization.h2.c": "3.0"
                    }
                }
            }
    },

        'mappings': {

            "properties":{
                "source":{
                    "type": "text",
                    "analyzer": "standard"
                    },
                "author":{
                    "type": "text" ,
                    "analyzer": "standard",
                    "similarity": "boolean"
                    },
                "title":{
                    "type": "text" ,
                    "analyzer": "english",
                    "similarity":"custom_dfr"
                    },

                "description":{
                    "type": "text" ,
                    "analyzer": "english",
                    "similarity":"custom_dfr"
                    },
                 "url":{
                    "type": "text"
                    },

                "publishedAt":{
                    "type" : "date"
                    },
                "timestamp" :{
                    "type" : "rank_feature",
                    "positive_score_impact" : True
                    }
                }
            }
        }
es.indices.create(index = index_name, body = request_body)
```

```
: {'acknowledged': True,
   'shards_acknowledged': True,
   'index': 'ms4976_info624_201904_newsproject4'}
```

- Out of the four similarities function which we have experimented with, the last similarity is performing well on our data. And we have applied this similarity to our primary index: **'ms4976_info624_201904_newsproject'**

- **Scoring:**

  We have used the rank_feature on our scoring/ranking of the documents which are being retrieved. We have applied the rank_feature on the field timestamp while constructing the primary index with the respective mappings.

- **Boosting Methods used: (Jupyter Notebook used is: Evaluation)**

  We have used the below-boosting queries on all the four indexes which we had defined for our similarities.

  At this stage, we have boosted the author, title, and source field on different similarities to evaluate the performance of our queries.

Use case 1 : "Crimes in US" on index "ms4976_info624_201904_newsproject1"

```
es.search(index = 'ms4976_info624_201904_newsproject1', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "crimes in US",
          "fields": ["title^2","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 2 : "articles by BBC news" on index "ms4976_info624_201904_newsproject1"

```
es.search(index = 'ms4976_info624_201904_newsproject1', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "articles by BBC news",
          "fields": ["source^3","title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 3 : "COVID-19" on index "ms4976_info624_201904_newsproject1"

```
es.search(index = 'ms4976_info624_201904_newsproject1', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "COVID-19",
          "fields": ["title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 4 : "Annie Karni articles on Trump" on index "ms4976_info624_201904_newsproject1"

```
es.search(index = 'ms4976_info624_201904_newsproject1', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "Annie Karni articles on Donald Trump",
          "fields": ["author^5","title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Evaluation on "ms4976_info624_201904_newsproject2" index which has different similarities configuration

Use case 1 : "US Elections" on index "ms4976_info624_201904_newsproject2"

```
]: es.search(index = 'ms4976_info624_201904_newsproject2', body=
   {"from":0, "size":10,
    "query":
    {
      "bool":
      {
        "must":
        [{
           "multi_match":
           {
             "query": "crimes in US",
             "fields": ["title^2","description"]
           }
        }],
        "should":
        {
          "rank_feature":
          {
            "field": "timestamp",
            "sigmoid":{"pivot": 5,"exponent":0.6}
          }
        }
      }
     }
   })
```

Use case 2 : "articles by BBC news" on index "ms4976_info624_201904_newsproject2"

```
es.search(index = 'ms4976_info624_201904_newsproject2', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "articles by BBC news",
          "fields": ["source^3","title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 3 : "COVID-19" on index "ms4976_info624_201904_newsproject2"

```
es.search(index = 'ms4976_info624_201904_newsproject2', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "COVID-19",
          "fields": ["title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 4 : "Annie Karni articles on Trump" on index "ms4976_info624_201904_newsproject2"

```
es.search(index = 'ms4976_info624_201904_newsproject2', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "Annie Karni articles on Donald Trump",
          "fields": ["author^5","title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Evaluation on "ms4976_info624_201904_newsproject3" index which has different similarities configuration

Use case 1 : "US Elections" on index "ms4976_info624_201904_newsproject3"

```
es.search(index = 'ms4976_info624_201904_newsproject3', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "crimes in US",
          "fields": ["title^2","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 2 : "articles by BBC news" on index "ms4976_info624_201904_newsproject3"

```
es.search(index = 'ms4976_info624_201904_newsproject3', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "articles by BBC news",
          "fields": ["source^3","title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 3 : "COVID-19" on index "ms4976_info624_201904_newsproject3"

```
es.search(index = 'ms4976_info624_201904_newsproject3', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "COVID-19",
          "fields": ["title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 4 : "Annie Karni articles on Trump" on index "ms4976_info624_201904_newsproject3"

```
es.search(index = 'ms4976_info624_201904_newsproject3', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "Annie Karni articles on Donald Trump",
          "fields": ["author^5","title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Evaluation on "ms4976_info624_201904_newsproject4" index which has different settings configuration

Use case 1 : "US Elections" on index "ms4976_info624_201904_newsproject4"

```
es.search(index = 'ms4976_info624_201904_newsproject4', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "crimes in US",
          "fields": ["title^2","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 2 : "articles by BBC news" on index "ms4976_info624_201904_newsproject4"

```
es.search(index = 'ms4976_info624_201904_newsproject4', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "articles by BBC news",
          "fields": ["source^3","title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 3 : "COVID-19" on index "ms4976_info624_201904_newsproject4"

```
es.search(index = 'ms4976_info624_201904_newsproject4', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "COVID-19",
          "fields": ["title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

Use case 4 : "Annie Karni articles on Trump" on index "ms4976_info624_201904_newsproject4"

```
es.search(index = 'ms4976_info624_201904_newsproject4', body=
{"from":0, "size":10,
 "query":
 {
   "bool":
   {
     "must":
     [{
        "multi_match":
        {
          "query": "Annie Karni articles on Donald Trump",
          "fields": ["author^5","title","description"]
        }
     }],
     "should":
     {
       "rank_feature":
       {
         "field": "timestamp",
         "sigmoid":{"pivot": 5,"exponent":0.6}
       }
     }
   }
  }
})
```

- Based on the above-boosting methods which we have used we will re-index our primary index for this project by boosting the fields: title^2, source^3, and author^5 to increase the weightage to find the relevance of the document during the query and normalization.

Index creation and mappings:

From the above, we have selected the 4th similarity function which we have tested and evaluated upon which we have received the most promising results.

We have followed the below steps for index creation and mapping:

1. Copy the data from the primary index to a temporary index.
2. Drop the primary index.
3. Create the same primary index with the selected similarities and the mappings as below.
4. Re-index the data from the temporary index to the primary index.

```
PUT /ms4976_info624_201904_newsproject/
{
"settings":{
        "index":{
          "similarity":{
            "custom_dfr":{
              "type": "DFR",
              "basic_model": "ine",
              "after_effect": "b",
              "normalization": "z",
              "normalization.h2.c": "3.0"
            }
          }
        }
}
}

PUT /ms4976_info624_201904_newsproject/_mapping
{
  "properties":{
          "source":{
            "type": "text",
            "analyzer": "standard"
            },
          "author":{
            "type": "text" ,
            "analyzer": "standard",
            "similarity": "boolean"
```

```
        },
      "title":{
        "type": "text" ,
        "analyzer": "english",
        "similarity":"custom_dfr"
        },

      "description":{
        "type": "text" ,
        "analyzer": "english",
        "similarity":"custom_dfr"
        },
       "url":{
        "type": "text"
        },

      "publishedAt":{
        "type" : "date"
        },
      "timestamp" :{
        "type" : "rank_feature",
        "positive_score_impact" : true
        }
      }
  }
```

• How good (Testing and Evaluation of Search Engine): **(Jupyter Notebook used is: Evaluation)**

Use Case 1:

The information needs for this query ("crimes in US") is to find all relevant news articles from the "ms4976_info624_201904_newsproject1" index which have any details regarding crimes that are happening in united states. We can observe from the query Top 10 results where Doc 1, Doc 5, Doc 8 are not relevant to our query information needs.

Below are relevant and non-relevant document based on the query information needs

Relevant = Doc 2, Doc 3, Doc 4, Doc 6, Doc 7, Doc 9, Doc 10

Non-Relevant = Doc 1, Doc 5, Doc 8

Precision is defined as percentage of retrieved docs that are relevant

Precision = True positive (TP)/True positive (TP) + False Positive (FP)

= 7/ (7+3)

=7/10

=0.7

DCG appears to have high value if the top retrieved documents are relevant

Discounted cumulative gain can be given by below formula:

DCG = rel1+ (rel2/log(2)) + (rel3/log(3)) + (rel4/log(4)) + (rel5/log(5)) + (rel6/log(6)) + (rel7/log(7)) +(rel8/log(8)) (rel9/log(9)) +(rel10/log(10))

Here relevance value for any document will be 1 if the retrieved document is relevant to the query needs and 0 if the retrieved document is not relevant to the query needs

DCG = 0+ (1/log(2)) + (1/log(3)) + (1/log(4)) + (0/log(5)) + (1/log(6)) + (1/log(7)) + (0/log(8)) + (1/log(9))+(1/log(10))

= 0 + 1/1 + 1/1.58 +1/2 + 0/2.32 + 1/2.58 + 1/2.8 + 0/3 + 1/3.17+ 1/3.32

= 1 + 0.63 + 0.5 + 0.39 +0.36 + 0.31 + 0.3

=3.49

IDCG can be calculated by re-ordering the retrieved results by making all relevant results at top in decreasing order of their scores

IDCG = 1+(1/log(2)) +(1/log(3)) + (1/log(4)) + (1/log(5)) + (1/log(6)) + (1/log(7)) +(0/log(8)) +(0/log(9)) +(0/log(10))

= 1 + 1/1 + 1/1.58 +1/2 + 1/2.32 + 1/2.58 + 1/2.8 + 0/3 + 0/3.17+ 0/3.32

= 1 + 1 + 0.63 + 0.5 +0.43 + 0.39 +0.36

=4.31

Normalized DCG (nDCG) can be calculated by using DCG and IDCG as shown below

nDCG = DCG/IDCG

=3.49/4.31

=0.8

Use case 2:

The information needs for this query ("articles by BBC news") is to find all relevant news articles from the "ms4976_info624_201904_newsproject1" index which are published by BBC news source. We can see from the query results that all the Top 10 retrieved results are relevant to our query information needs.

Below are relevant and non-relevant document based on the query information needs

relevant = Doc1, Doc 2, Doc 3, Doc 4, Doc 5, Doc 6, Doc 7, Doc 8, Doc 9, Doc 10

non-relevant = Null

For this query, we are getting no False Positive as all the retrieved results are relevant to the information needs
Precision is defined as percentage of retrieved docs that are relevant

Precision = True positive (TP)/True positive (TP) + False Positive (FP)

= 10/ (10+0)

=10/10

=1

DCG appears to have high value if the top retrieved documents are relevant

Discounted cumulative gain can be given by below formula:

DCG = rel1+ (rel2/log(2)) + (rel3/log(3)) + (rel4/log(4)) + (rel5/log(5)) + (rel6/log(6)) + (rel7/log(7)) + (rel8/log(8)) (rel9/log(9)) +(rel10/log(10))

Here relevance value for any document will be 1 if the retrieved document is relevant to the query needs and 0 if the retrieved document is not relevant to the query needs

DCG = 1+ (1/log(2)) + (1/log(3)) + (1/log(4)) + (1/log(5)) + (1/log(6)) + (1/log(7)) + (1/log(8)) + (1/log(9)) +(1/log(10))

= 1 + 1/1 + 1/1.58 +1/2 + 1/2.32 + 1/2.58 + 1/2.8 + 1/3 + 1/3.17+ 1/3.32

= 1 + 1 + 0.63 + 0.5 + 0.43 + 0.39 + 0.36 + 0.33 + 0.31 + 0.3

=5.25

IDCG can be calculated by re-ordering the retrieved results by making all relevant results at top in decreasing order of their scores

IDCG = 1+ (1/log(2)) + (1/log(3)) + (1/log(4)) + (1/log(5)) + (1/log(6)) + (1/log(7)) + (1/log(8)) + (1/log(9)) +(1/log(10))

= 1 + 1/1 + 1/1.58 +1/2 + 1/2.32 + 1/2.58 + 1/2.8 + 1/3 + 1/3.17+ 1/3.32

= 1 + 1 + 0.63 + 0.5 + 0.43 + 0.39 +0.36 + 0.33 + 0.31 + 0.3

=5.25

Normalized DCG (nDCG) can be calculated by using DCG and IDCG as shown below

nDCG = DCG/IDCG

=5.25/5.25

=1

Use case 3:

The information needs for this query "COVID-19" is to find all relevant news articles from the "ms4976_info624_201904_newsproject1" index which are related to covid -19 virus topics. We can see from the query results that all the Top 10 retrieved results are relevant to our query information needs.

Below are relevant and non-relevant document based on the query information needs

relevant = Doc1, Doc 2, Doc 3, Doc 4, Doc 5, Doc 6, Doc 7, Doc 8, Doc 9, Doc 10

non-relevant = Null

For this query, we are getting no False Positive as all the retrieved results are relevant to the information needs.
Precision is defined as percentage of retrieved docs that are relevant

Precision = True positive (TP)/True positive (TP) + False Positive (FP) = 10/ (10+0) =10/10 =1

DCG appears to have high value if the top retrieved documents are relevant

Discounted cumulative gain can be given by below formula:

DCG = rel1+ (rel2/log(2)) + (rel3/log(3)) + (rel4/log(4)) + (rel5/log(5)) + (rel6/log(6)) + (rel7/log(7)) + (rel8/log(8)) (rel9/log(9)) +(rel10/log(10))

Here relevance value for any document will be 1 if the retrieved document is relevant to the query needs and 0 if the retrieved document is not relevant to the query needs

DCG = 1+ (1/log(2)) + (1/log(3)) + (1/log(4)) + (1/log(5)) + (1/log(6)) + (1/log(7)) + (1/log(8)) + (1/log(9)) +(1/log(10))

= 1 + 1/1 + 1/1.58 +1/2 + 1/2.32 + 1/2.58 + 1/2.8 + 1/3 + 1/3.17+ 1/3.32

= 1 + 1 + 0.63 + 0.5 + 0.43 + 0.39 + 0.36 + 0.33 + 0.31 + 0.3

=5.25

IDCG can be calculated by re-ordering the retrieved results by making all relevant results at top in decreasing order of their scores

IDCG = 1+ (1/log(2)) + (1/log(3)) + (1/log(4)) + (1/log(5)) + (1/log(6)) + (1/log(7)) + (1/log(8)) + (1/log(9)) +(1/log(10))

= 1 + 1/1 + 1/1.58 +1/2 + 1/2.32 + 1/2.58 + 1/2.8 + 1/3 + 1/3.17+ 1/3.32

= 1 + 1 + 0.63 + 0.5 + 0.43 + 0.39 +0.36 + 0.33 + 0.31 + 0.3

=5.25

Normalized DCG (nDCG) can be calculated by using DCG and IDCG as shown below

nDCG = DCG/IDCG

=5.25/5.25

=1

Use case 4:

The information needs for this query ("Annie Karni articles on Donald Trump") is to find all relevant news articles from the "ms4976_info624_201904_newsproject1" index which are Published by Annie Karni Author on any Donald Trump topics. We can observe from the query Top 10 results where Doc 1 is not relevant to our query information needs.

Based on the above query information needs we can categorize these results into relevant and non-relevant document

relevant = Doc 2, Doc 3, Doc 4, Doc 5, Doc 6, Doc 7, Doc 8, Doc 9, Doc 10

non-relevant = Doc 1

For this query, we are getting only one False Positive document among top 10 retrieved results and remaining all are relevant to the information needs.
Precision is defined as percentage of retrieved docs that are relevant

Precision = True positive (TP)/True positive (TP) + False Positive (FP)

= 9/ (9+1)

=9/10

=0.9

DCG appears to have high value if the top retrieved documents are relevant

Discounted cumulative gain can be given by below formula:

DCG = rel1+ (rel2/log(2)) + (rel3/log(3)) + (rel4/log(4)) + (rel5/log(5)) + (rel6/log(6)) + (rel7/log(7)) + (rel8/log(8)) (rel9/log(9)) +(rel10/log(10))

Here relevance value for any document will be 1 if the retrieved document is relevant to the query needs and 0 if the retrieved document is not relevant to the query needs

DCG = 0+ (1/log(2)) + (1/log(3)) + (1/log(4)) + (1/log(5)) + (1/log(6)) + (1/log(7)) + (1/log(8)) + (1/log(9)) +(1/log(10))

= 0 + 1 + 0.63 + 0.5 + 0.43 + 0.39 + 0.36 + 0.33 + 0.31 + 0.3

= 4.25

IDCG can be calculated by re-ordering the retrieved results by making all relevant results at top in decreasing order of their scores

IDCG = 1+ (1/log(2)) + (1/log(3)) + (1/log(4)) + (1/log(5)) + (1/log(6)) + (1/log(7)) + (1/log(8)) + (1/log(9)) +(0/log(10))

= 1 + 1/1 + 1/1.58 +1/2 + 1/2.32 + 1/2.58 + 1/2.8 + 1/3 + 1/3.17+ 0/3.32

= 1 + 1 + 0.63 + 0.5 + 0.43 + 0.39 +0.36 + 0.33 + 0.31 + 0

=4.95

Normalized DCG (nDCG) can be calculated by using DCG and IDCG as shown below

nDCG = DCG/IDCG

=4.25/4.95

=0.85

- We have done the above calculations for all the four indexes which we have used for selecting similarity.

- We have used the Evaluation jupyter notebook where we have shown in more detailed steps all the calculations.

Results:

- **Comparison of all Indices:**

Comparision of all Indices

Index	usecase 1 precision	usecase 1 nDCG	usecase 2 precision	usecase 2 nDCG	usecase 3 precision	usecase 3 nDCG	usecase 4 precision	usecase 4 nDCG	Average precision
ms4976_info624_201904_newsproject1	0.7	0.8	1	1	1	1	0.9	0.85	0.9
ms4976_info624_201904_newsproject2	0.7	0.8	1	1	1	1	0.7	0.61	0.85
ms4976_info624_201904_newsproject3	0.7	0.8	1	1	1	1	0.6	0.54	0.83
ms4976_info624_201904_newsproject4	0.7	0.8	1	1	1	1	1	1	0.93

- In Evaluating our index, we are much focused on precision and nDCG metrics rather than recall metric as this is not important in our domain objective. We can't judge our search engine precision only by taking consideration of one query results, so we have considered different use cases and we are taking average precision and nDCG values to evaluate our index performance.

- For all Indices, We obtained almost same precision results for most of the queries/use cases as our data collection is very small when compared to real time situations due to this our results has no significant difference and by considering the nDCG metric evaluations the index "ms4976_info624_201904_newsproject4" is giving better results compared to other indices and as we more interested on the top relevant results than lower order relevant results we have chosen this index as best in our case.

• Where:

Sr. No	Index Name	
1	ms4976_info624_201904_newsproject	Main Index
2	ms4976_info624_201904_newsproject1	Testing index 1
3	ms4976_info624_201904_newsproject2	Testing index 2
4	ms4976_info624_201904_newsproject3	Testing index 3
5	ms4976_info624_201904_newsproject4	Testing index 4

• Experiences:

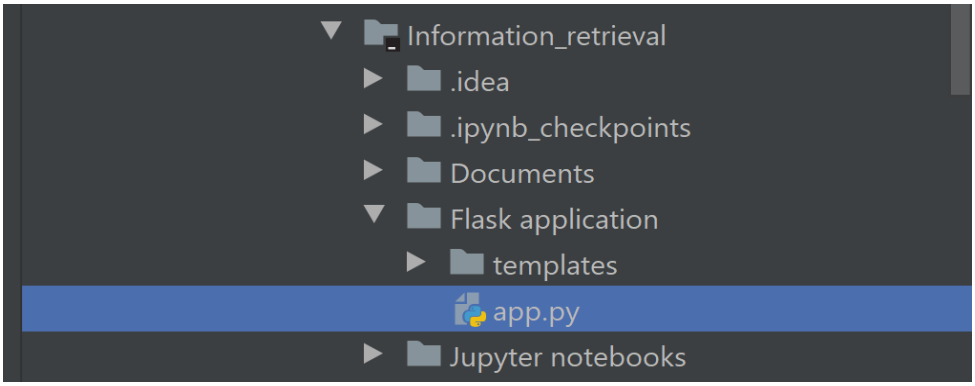
When we initially started to think about building a leading search engine, there were many thoughts and fundamental ideas which each one of us came up with. We then considered building a search engine that could retrieve relevant, readable, and effective search. We had to find a specific topic to build our search engine which would serve as our domain for the data collection. The very subsequent challenge in common is to check the feasibility of it, finding out the relevant, informative data source, we obtained many options to be considered, and later after doing quite a bit of research we came across the “NewsAPI”, which we could get the API endpoint to be used within our system. Huge data handling and finding the relevant data set took us a while. It supported us to understand how to access the data through certain API and then fetched the data. While the documents were residing in the “Kibana” server, we tried to upload as much as possible of around twenty thousand records and this is when we realized and got an insight that it isn't easy to handle enormous data and then we had to maintain the data within ten thousand which we were capable to succeed. In view of loading the data, we have used python to upload the data which was restricted within twenty documents however through the script we succeeded to upload more than twenty. Acquiring such enormous data and trying to have a query with respect to the relevance with the retrieved data remain a challenging task to understand the different probabilistic models which gained us some knowledge in querying. From this project, we have understood the concepts of Custom Similarities and how to define them, which similarity works best for our data. The concept of boosting the fields and how the weightage of the keywords changed with respect to the relevance of the data.

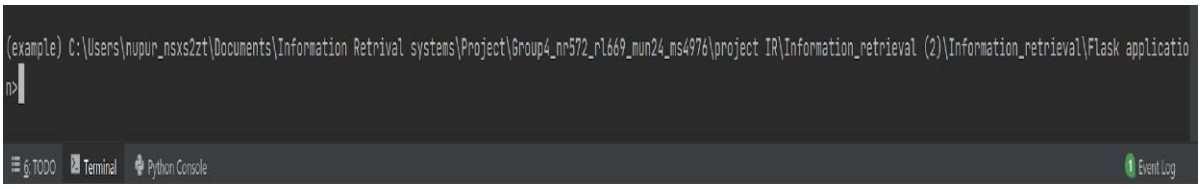
0.2 Code and Interface:

Steps to execute the code and Interface:

1. Install python with the latest version.

2. Change the directory to the folder where we have the file called: app.py





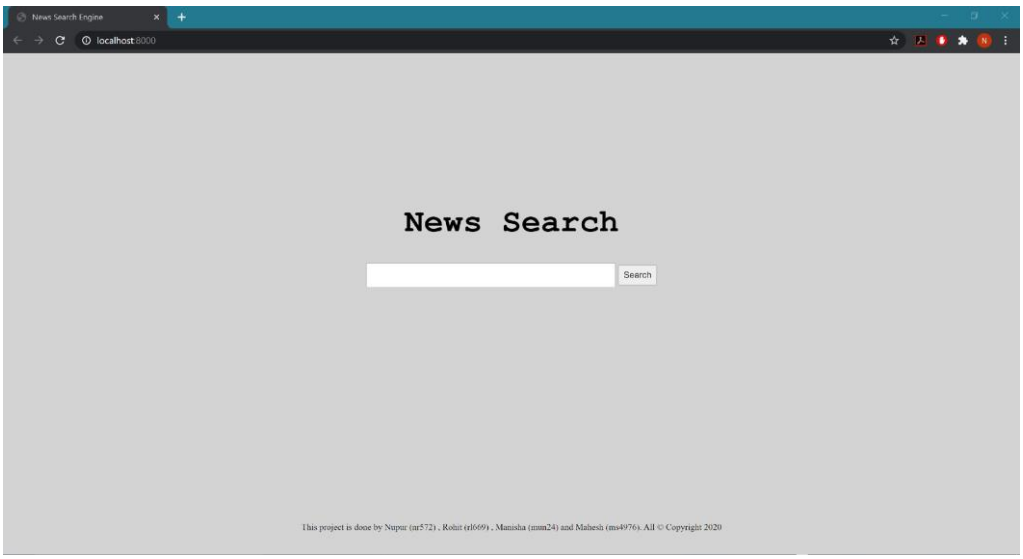
3. Install all the below-mentioned modules using the command “pip install module_name” while being in the same location.
    - flask
    - elasticsearch
    - requests

4. Run the flask application using the command “python app.py “

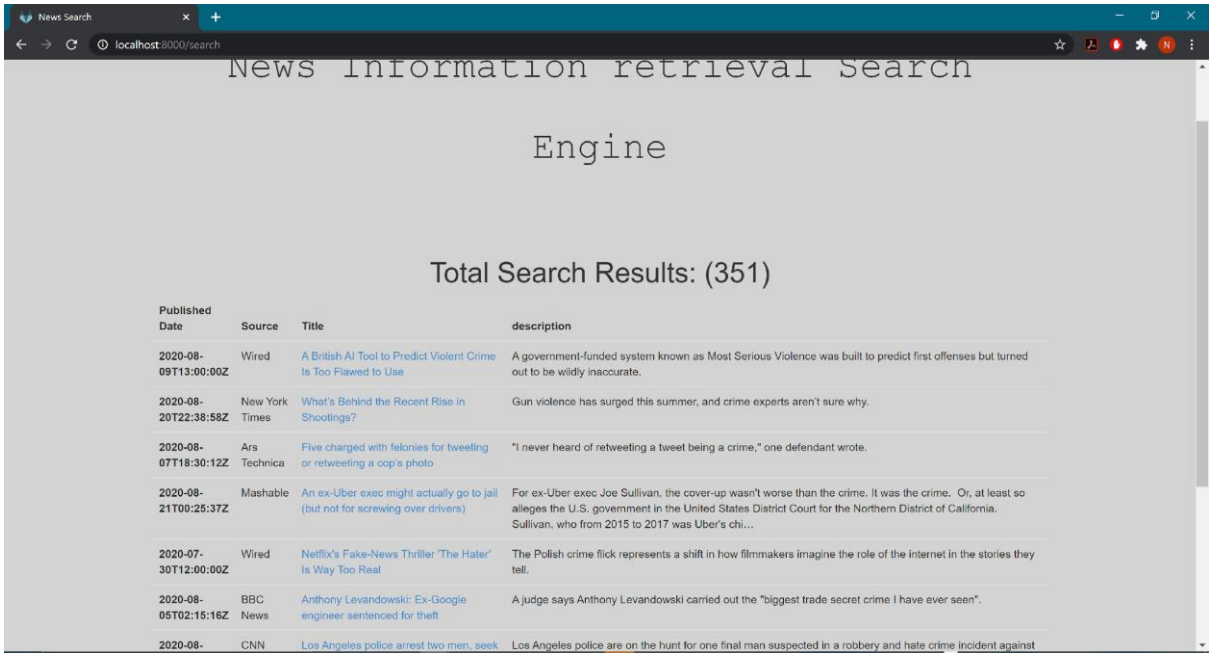
5. Now, the flask application should start running on the local server on the specified port. (we have specified port:8000)

6. Click on the ‘URL’ generated: http://localhost:8000/

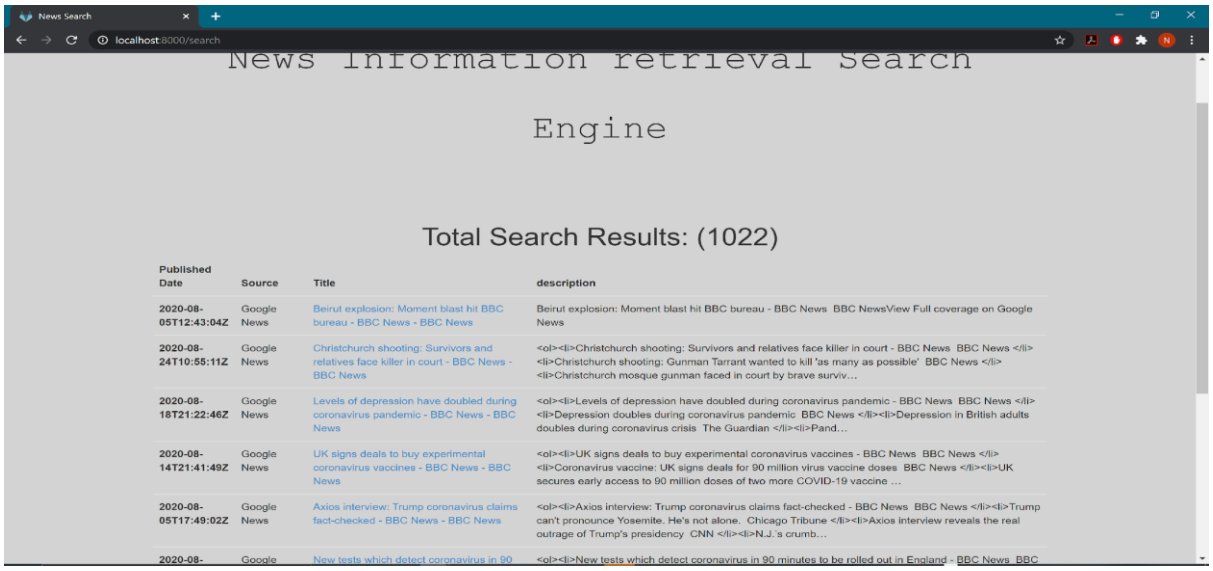
UI Screenshots:



Use Case 1:

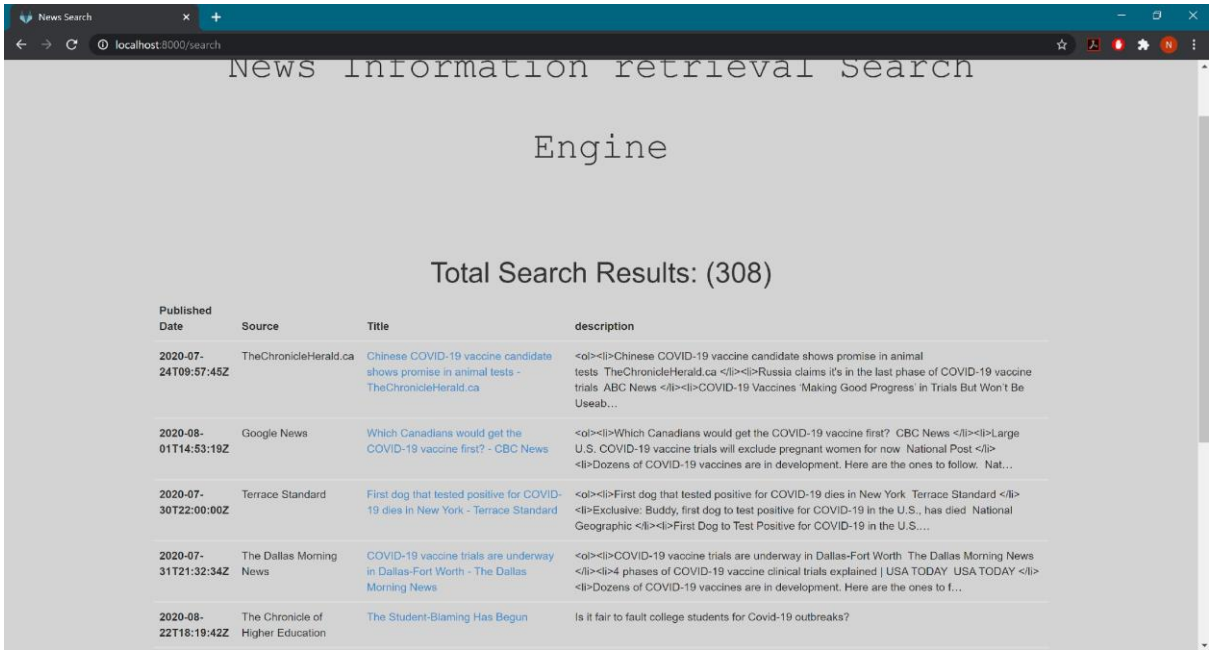


Use case 2:

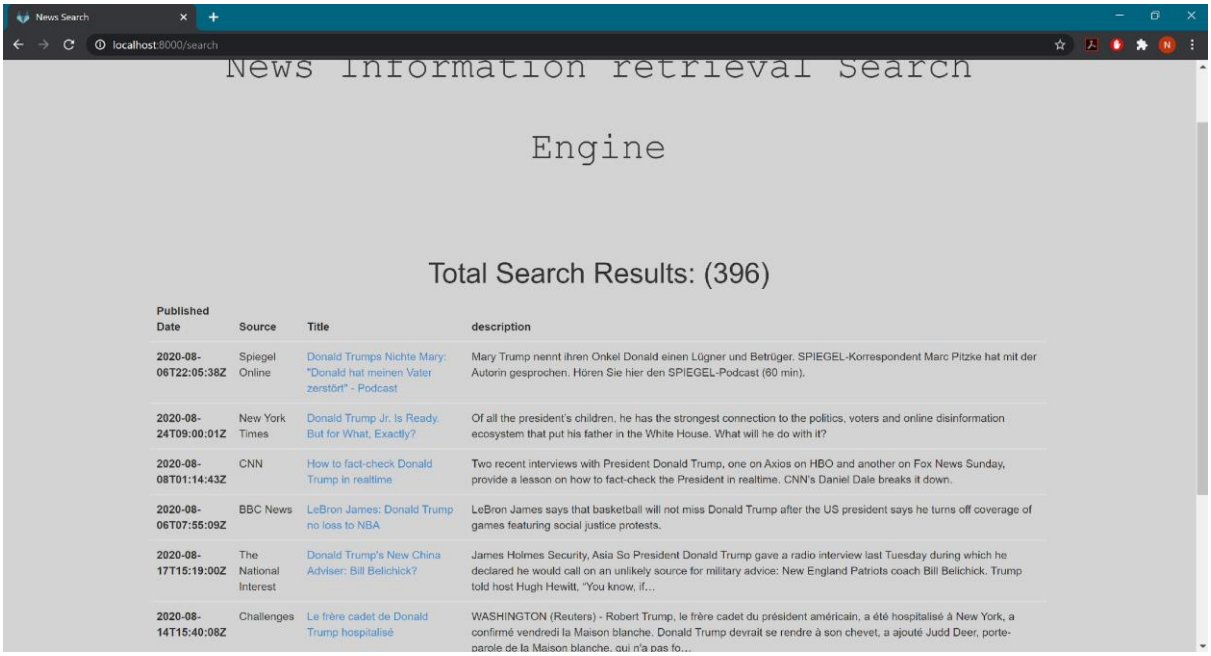

News Search
localhost:8000/search
News Information retrieval Search
Engine
Total Search Results: (1022)
Published Date
Source
Title
description
2020-08-05T12:43:04Z
Google News
Beirut explosion: Moment blast hit BBC bureau - BBC News - BBC News
Beirut explosion: Moment blast hit BBC bureau - BBC News BBC NewsView Full coverage on Google News
2020-08-24T10:55:11Z
Google News
Christchurch shooting: Survivors and relatives face killer in court - BBC News - BBC News
Christchurch shooting: Survivors and relatives face killer in court - BBC News BBC News Christchurch shooting: Gunman Tarrant wanted to kill 'as many as possible' BBC News Christchurch mosque gunman faced in court by brave surviv…
2020-08-18T21:22:46Z
Google News
Levels of depression have doubled during coronavirus pandemic - BBC News - BBC News
Levels of depression have doubled during coronavirus pandemic - BBC News BBC News Depression doubles during coronavirus pandemic BBC News Depression in British adults doubles during coronavirus crisis The Guardian Pand…
2020-08-14T21:41:49Z
Google News
UK signs deals to buy experimental coronavirus vaccines - BBC News - BBC News
UK signs deals to buy experimental coronavirus vaccines - BBC News BBC News Coronavirus vaccine: UK signs deals for 90 million virus vaccine doses BBC News UK secures early access to 90 million doses of two more COVID-19 vaccine …
2020-08-05T17:49:02Z
Google News
Axios interview: Trump coronavirus claims fact-checked - BBC News - BBC News
Axios interview: Trump coronavirus claims fact-checked - BBC News BBC News Trump can't pronounce Yosemite. He's not alone. Chicago Tribune Axios interview reveals the real outrage of Trump's presidency CNN N.J.'s crumb…
2020-08-
Google
New tests which detect coronavirus in 90
New tests which detect coronavirus in 90 minutes to be rolled out in England - BBC News BBC

Use case 3:


News Search
COVID-19
Search


News Search
localhost:8000/search
News Information retrieval Search
Engine
Total Search Results: (308)
Published Date
Source
Title
description
2020-07-24T09:57:45Z
TheChronicleHerald.ca
Chinese COVID-19 vaccine candidate shows promise in animal tests - TheChronicleHerald.ca
Chinese COVID-19 vaccine candidate shows promise in animal tests TheChronicleHerald.ca Russia claims it's in the last phase of COVID-19 vaccine trials ABC News COVID-19 Vaccines 'Making Good Progress' in Trials But Won't Be Useab…
2020-08-01T14:53:19Z
Google News
Which Canadians would get the COVID-19 vaccine first? - CBC News
Which Canadians would get the COVID-19 vaccine first? CBC News Large U.S. COVID-19 vaccine trials will exclude pregnant women for now National Post Dozens of COVID-19 vaccines are in development. Here are the ones to follow. Nat…
2020-07-30T22:00:00Z
Terrace Standard
First dog that tested positive for COVID-19 dies in New York - Terrace Standard
First dog that tested positive for COVID-19 dies in New York Terrace Standard Exclusive: Buddy, first dog to test positive for COVID-19 in the U.S., has died National Geographic First Dog to Test Positive for COVID-19 in the U.S.…
2020-07-31T21:32:34Z
The Dallas Morning News
COVID-19 vaccine trials are underway in Dallas-Fort Worth - The Dallas Morning News
COVID-19 vaccine trials are underway in Dallas-Fort Worth The Dallas Morning News 4 phases of COVID-19 vaccine clinical trials explained | USA TODAY USA TODAY Dozens of COVID-19 vaccines are in development. Here are the ones to f…
2020-08-22T18:19:42Z
The Chronicle of Higher Education
The Student-Blaming Has Begun
Is it fair to fault college students for Covid-19 outbreaks?

Use case 4:



0.3 References:

References and Citations taken from:

1. https://sysadmins.co.za/building-a-search-engine-for-our-scraped-data-on-elasticsearch-part-2/
2. https://newsapi.org/docs/endpoints/top-headlines

0.4 Future Scoping:

- The limitation of the index size in the ElasticServers.
- Generating the data retrieved from the API dynamically.
- Adding pagination to the results page.