University of Essex

Department of Mathematical Sciences

CE706 Information Retrieval Assignment 2
Elasticsearch and Evaluation

Students name

BAKSHI, CHANDA : 1907683

PANDEY, NIRUPMA: 1906463

Date of submission :April 9, 2020

Contents

1 Introduction

With a rapid increase in digital marketing and online recommendation systems, B2B software service providers have generated the need for Search Engine Optimization (SEO), to attract target customers, boost their authority and credibility and stay ahead of similar market competitors. In this assignment, we aimed to engineer different approaches to SEO and to evaluate their performances. We are given a collection of articles, downloaded from 'The Signal Media One-Million News Articles Dataset[1] into a JSON line file and imported on Kibana for building indexing and searching software.

The first Web information services were based on traditional information retrieval (IR) algorithms and techniques. However, IR algorithms were developed for smaller and more coherent collections than the Web is. Thus Web searching requires new techniques - exploiting linkage among Web pages or extensions of the old ones, for example[2]. Discovery of smartphones, better internet speed, increasing social media craze, online or digital marketing has made people's life easy and now a majority of world population relying on web browsing to retrieve information. The indexed web is readily available to the public and searchable with standard web search engines, but generally not cover all relevant information causing deficiencies like non-flexibility in search and result in least-precision pieces of information. Hence all these have caused a need for search engine optimisation, in order to retrieve more precise information that the user is looking for.

Figure 1: Benefits of a Search Engine Optimization

2 Software installation

In this section we are going to look at the software required for this task and going to discuss about procedure we used by us for their installation:

2.1 Elasticsearch

Elasticsearch is Lucene library based open source search engine which was developed on JAVA and It provides a multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.To start with our analysis we have downloaded and installed Elasticsearch 7.6.2 version and run “bin\elasticsearch.bat” query in the powershell. Figure-2 shows running of the elasticsearch.

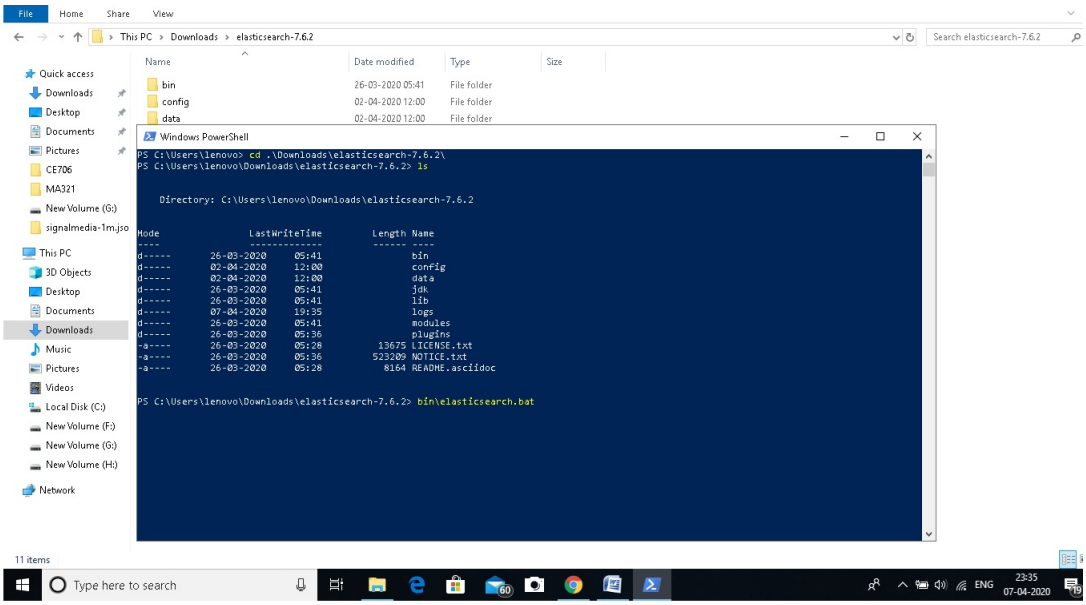


Figure 2: Running Elasticsearch after installation

Once Elasticsearch is installed and running in the system, then go to the browser and type “http://localhost:9200” to check the cluster name, installed version and all the details as shown in figure-3.

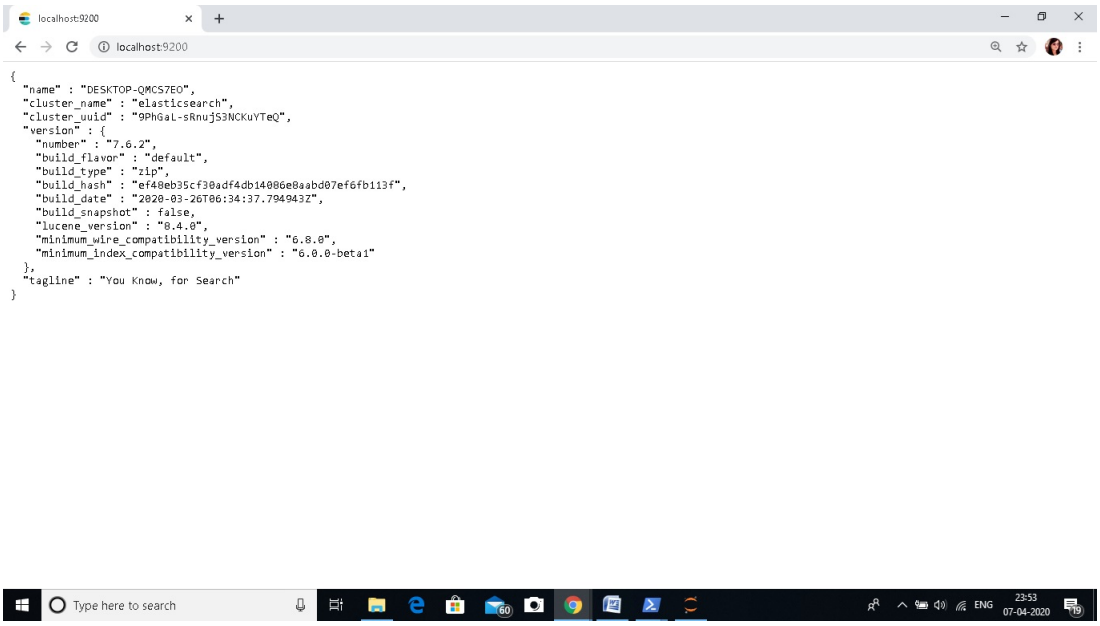


Figure 3: Checking localhost name localhost:9200

2.2 Kibana

Kibana is an open source data visualization dashboard using bar, line and scatter plots, or pie charts and maps on top of large volumes of data for Elasticsearch, used for content indexed on a cluster.We have downloaded and installed Kibana 7.6.2 version and run “bin\kibana.bat” query in the powershell, as shown in below figure.

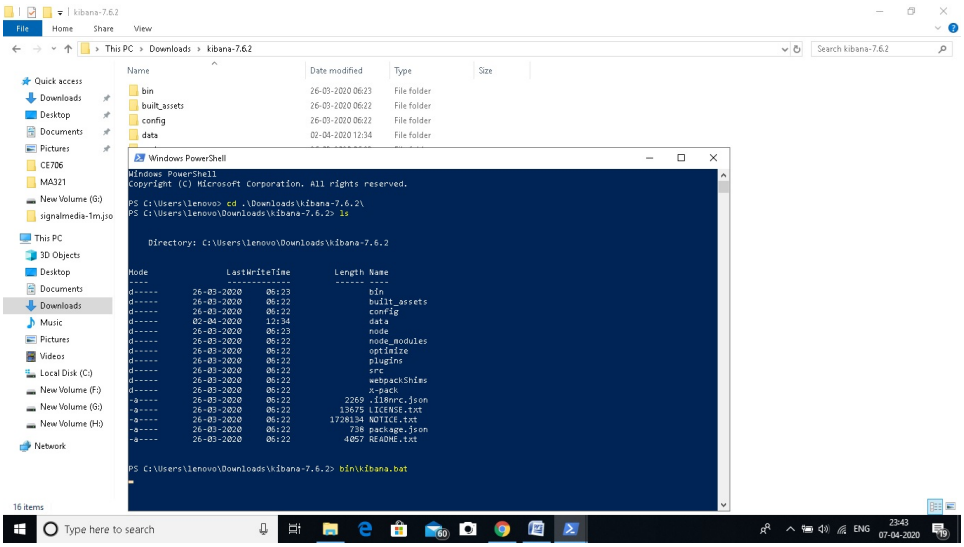


Figure 4: Running Elasticsearch after installation

Once Kibana is installed and running in the system, then go to the browser and type “http://localhost:5601” to check the cluster name, installed version and all the other details as shown in figure-5.

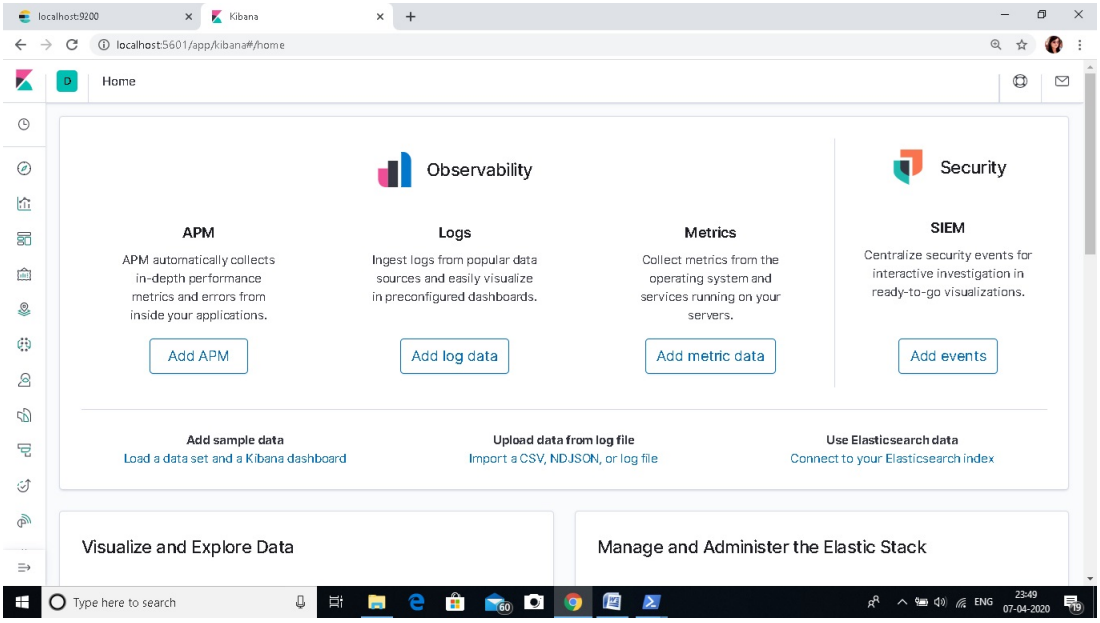


Figure 5: Checking localhost name localhost:5601

3 Handling Dataset

We are given with a collection of articles, downloaded from ”https://research.signal-ai.com/newsir16/signal-dataset.html”, ‘The Signal Media One-Million News Articles dataset’ into JSON line file format, where each line is a JSON object representing an article. Each of the article contains below fields:

- id: An unique identifier for each article
- title: Given title of the article
- content: The textual content of the article
- source: Source name for of the article (e.g. Reuters)
- published: The date of publication of the article
- media-type: Categories wither a ”News” or a ”Blog”

In python version 3.6.7, using below mentioned libraries to import the JSON file on Kibana

- sys: for system-specific parameter and functions
- requests: HTTP library for importing data from HTML
- json: to work on JSON format file

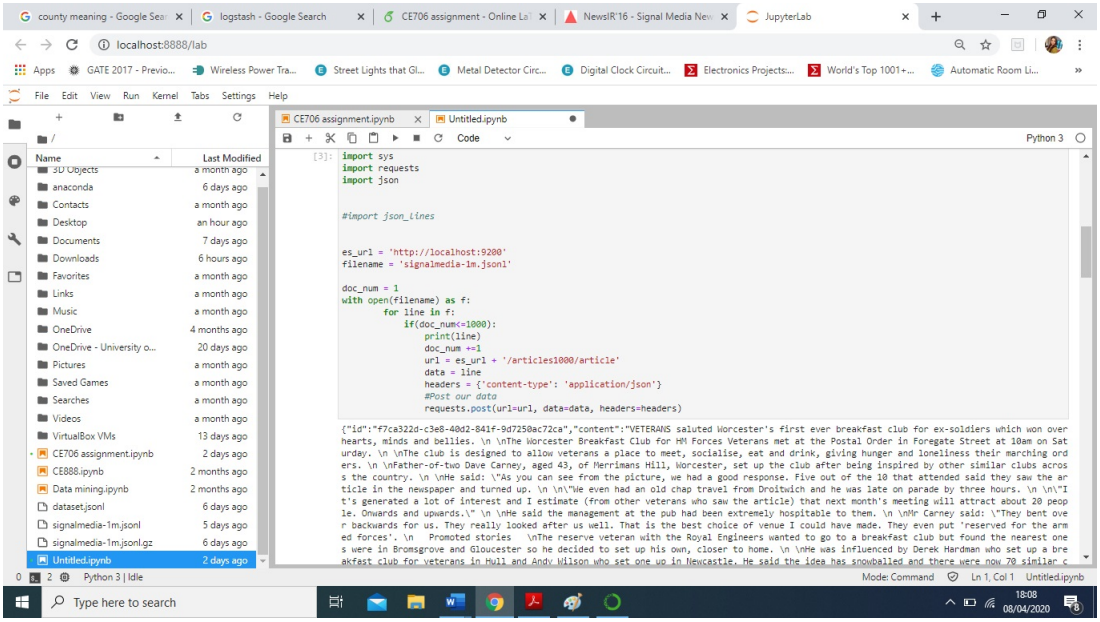


Figure 6: Screenshot showing python codes used for importaing data from HTML to Kibana

4 Indexing

In this section we have loaded the dataset on python version 3 and defined functions to create indexes, which is described in below mentioned steps:

- Pointing to elasticsearch, initializing port
- Defining function indexing() to read and index JSON line format
- function json_lines.reader() to read data from JSONL
- json.dumps() converting string formatted data to json file format
- json.loads() reading the file in order to load on elasticsearch
- es.index() to add index, using four parameters index name, doc type, document number and the query.

All the above coding has been done in ”Assignment2.py” file and the index has been successfully loaded on Kibana. Now we can see the uploaded indexes in Kibana as Management –>Index Management

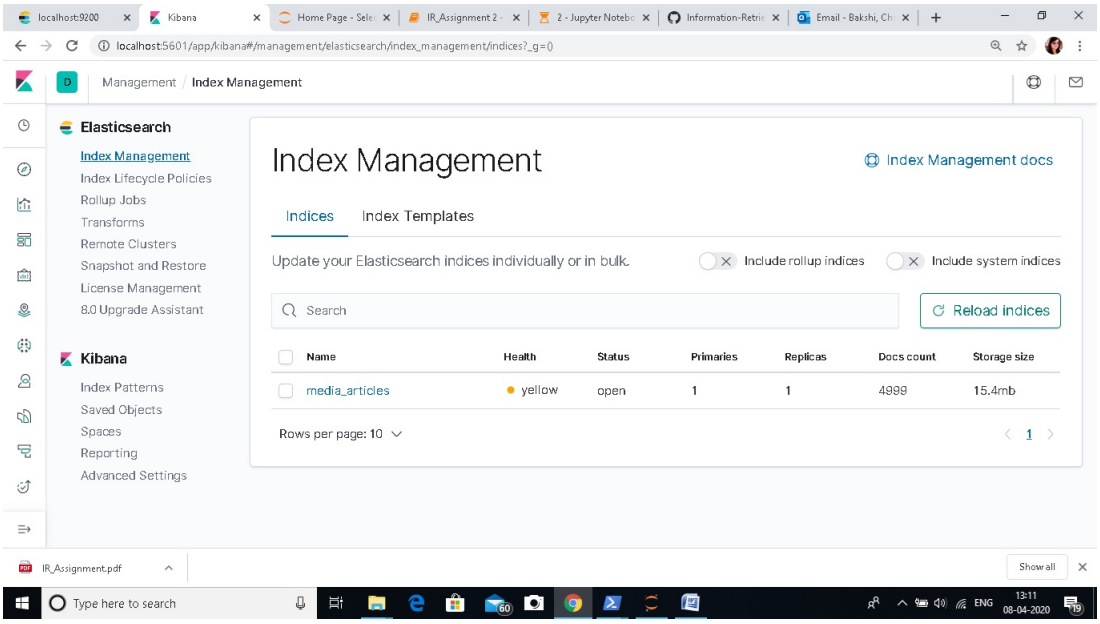


Figure 7: Screenshot showing Kibana dashboard

A default mapping has been created while loading the data on the Elasticsearch, names as ’media_article’ under the mapping section, as shown on figure 8 below.

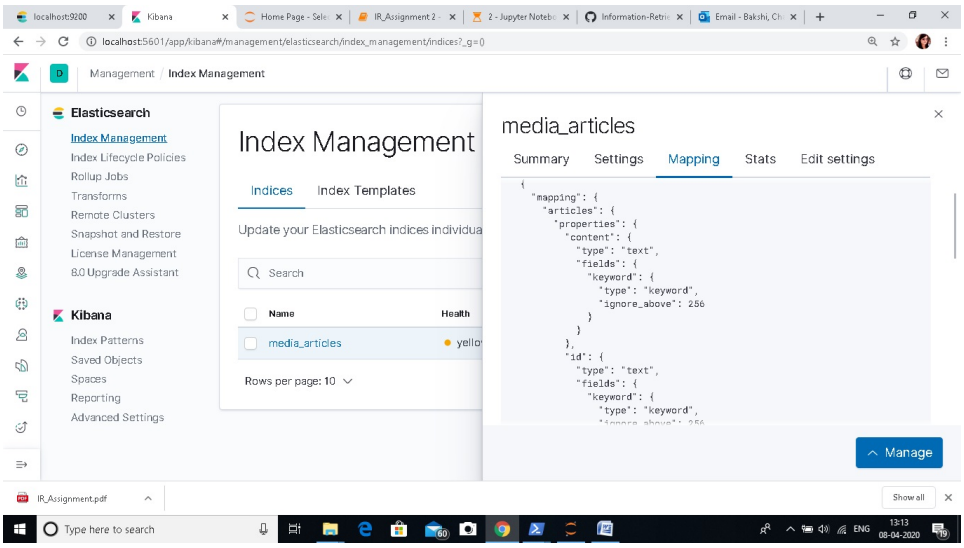
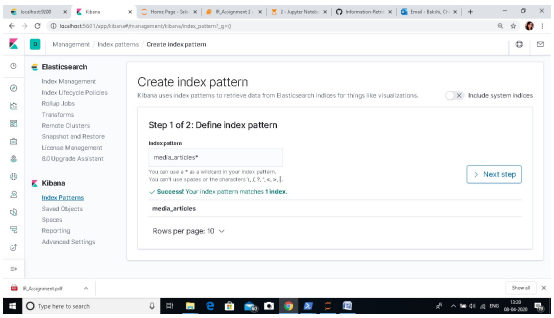


Figure 8: Index mapping



(a) Showing step 1 of index creation

(b) Showing step 2 of index creation

Figure 9: Creating a new index under the tab Create Index Pattern

For creating a new index go to tab Management –>Index Management–>Create Index Pattern as shown in figure 9.

5 Searching

Now, the index has been created under the Create Index Patter tab, with the name of 'media_article', we are able to search files on the Discover tab. Our dataset is containing mentioned fields,_id, _index, _source, _score, type, content, id, media–type, published, source, title.

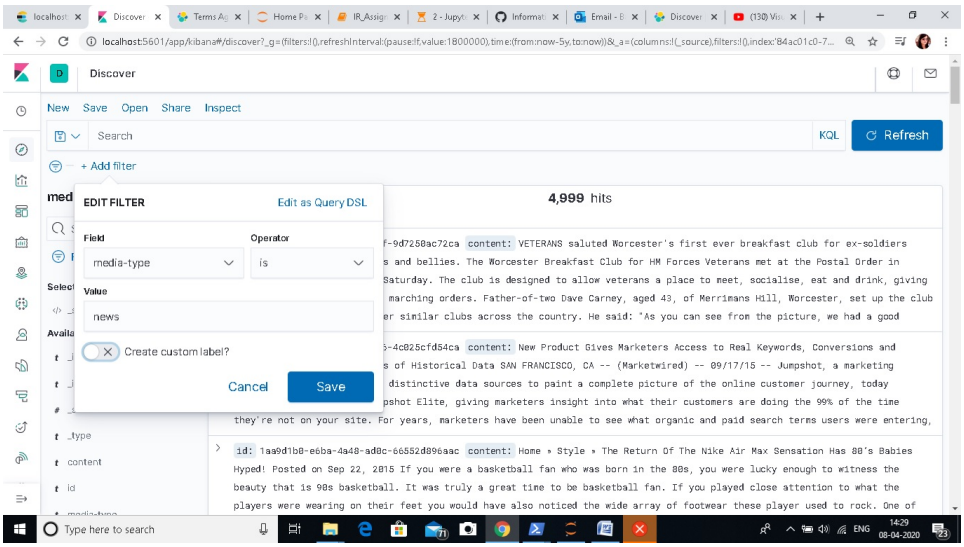


Figure 10: Field-wise search with media–type

Since our dataset is containing 11 fields, we can apply one or multiple filter to search for a specific article. Click on Add Filter, then select filed type you want to search and save changes, will show you desirable outputs. Figure 11 showing output after applying filter with media–type.

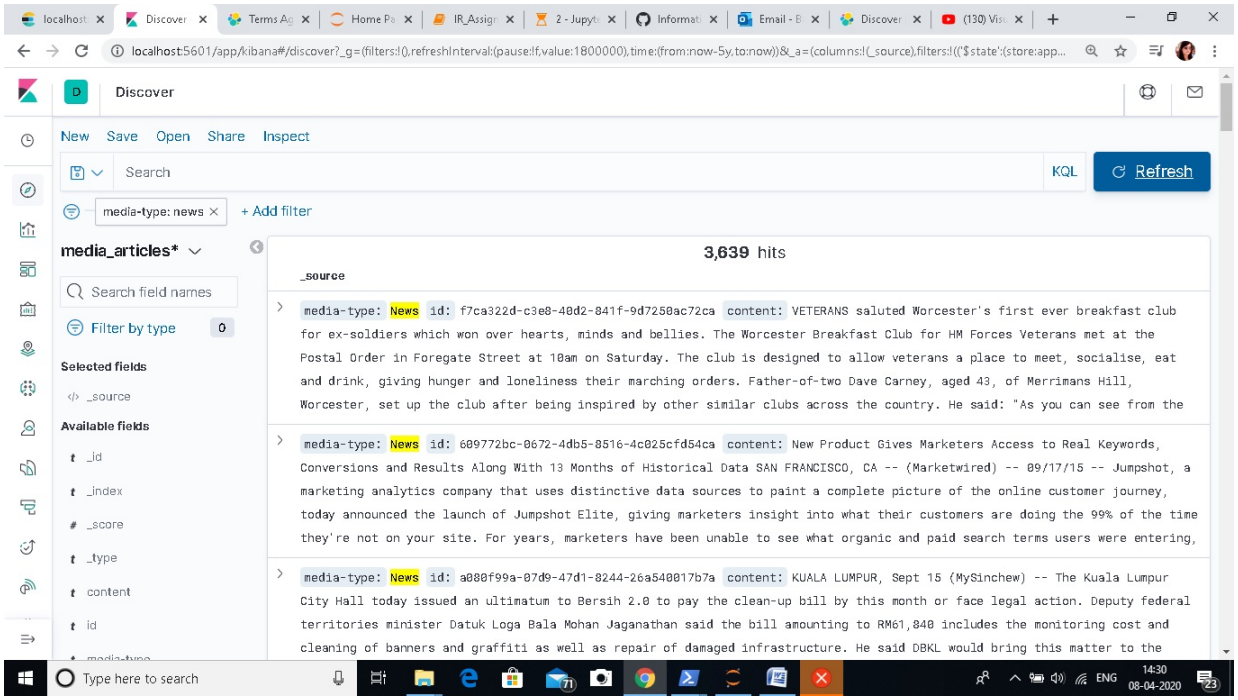


Figure 11: Search output for media–type field

In the filter, we can pick a field for example source and then specify a source 'Brisbane Times' to get specific result, as shown in below figures 12 and 13.

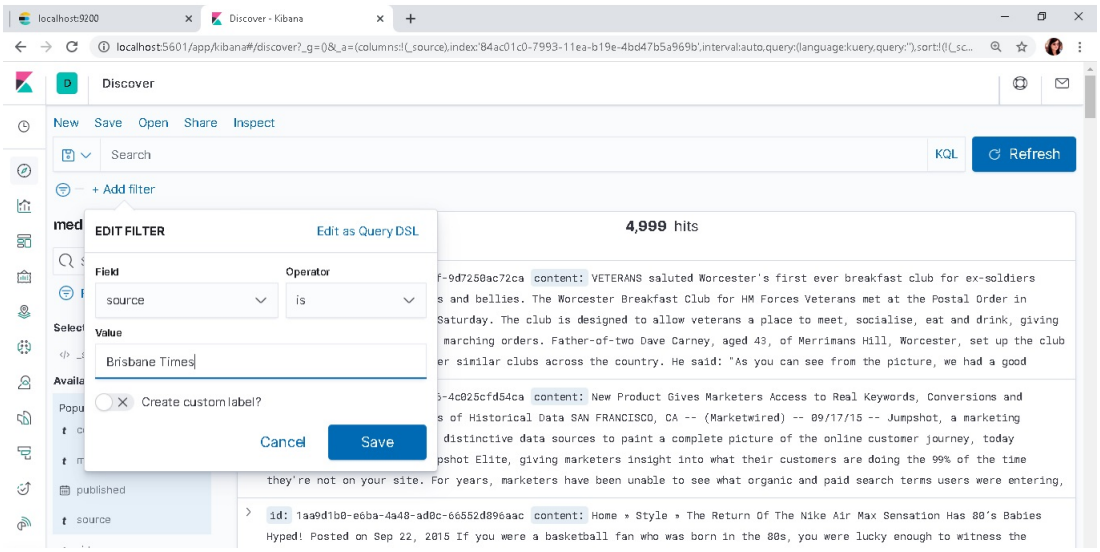


Figure 12: Applying source wise field search filter

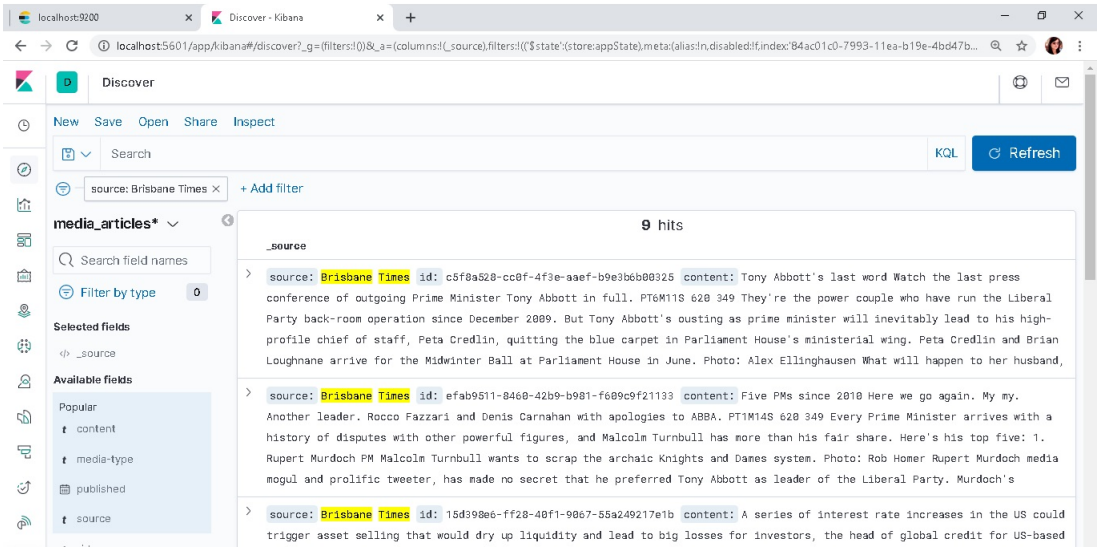


Figure 13: Specifying Brisbane Times as source

Adding another filter with a specific content Prime Minister, shown in figure

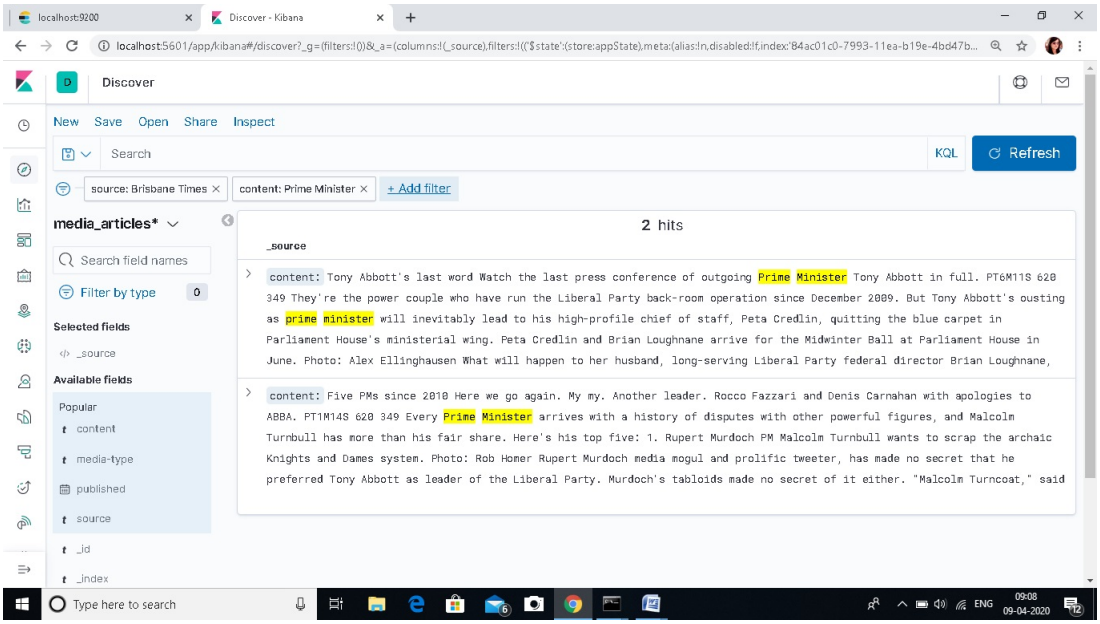


Figure 14: Adding another filter content: Prime minister

6 Test Samples creation

We have defined function query_search() with one parameter, asking the user to enter input for field-wise search. In this function, we have defined 10 different combinations of filters can be applied to a generate query, as mentioned below:

- Query 1: This will search with title and published date

  title=input('Enter the Title:')

  start=input('Enter the Published Start Date(YYYY/MM/DD):')

  end=input('Enter the Published End Date(YYYY/MM/DD):')

  Output: res = (index="media_articles", doc_type="articles", body="query":"bool":"should":["match":"title"

  start,"lte": end,"format": "yyyy/MM/dd——yyyy"],size=5000)

- Query 2 : This will search with title and article content

  title=input('Enter the Title:')

  content=input('Enter the content phrase:')

  Output: res= (index="media_articles",doc_type="articles",body="query":"bool":"should":["match":"title":

- Query 3: This will search with media type, content and published date

  media=input('Enter the Media-type:')

  content=input('Enter the content phrase:')

  start=input('Enter the Published Start Date(YYYY/MM/DD):')

  end=input('Enter the Published End Date(YYYY/MM/DD):')

  Output: res = (index="media_articles", doc_type="articles", body="query":"bool":"should":["match":"med

  type":media,"match":"content":content,"range":"published":"gte": start,"lte": end,"format":

  "yyyy/MM/dd——yyyy"],size=5000)

- Query 4: This will search with title and content

  title=input('Enter the Title:')

  content=input('Enter the content start phrase:')

  Output: res = (index="media_articles",doc_type="articles",body="query":"bool":"should":["match":"title":

$"content" : content], size = 5000)Query5 : Thiswillsearchwithtitle$

$title = input('EntertheExactTitle :')$

$Output : res = (index = "media\_articles", doc\_type = "articles", body = "query" : "match_phrase" :$

$5000)$

- Query 6: This will search with keyword

  keyword=input('Enter Query Keyword:')

  Output: res = (index="media_articles",doc_type="articles",body="query": "multi$_{m}atch$" :

  $"query" : keyword, "fields" : ["content^2", "title"], size = 5000)Query7 : Thiswillsearchwitharticle$

  $content = input("entertheletter : ")$

  $content = content + " * "$

  $Output : res = (index = "media\_articles", doc\_type = "articles", body = "query" : "wildcard" : "co$

  $5000)$

- Query 8: This will search with source and article content

  source=input('Enter the Source:')

  content=input('Enter the content phrase:')

  Output: res = (index="media_articles",doc_type="articles",body="query":"bool":"should":["match":"sourc

- : Query 9: This will search with query

  query=input('Enter the query:')

  Output: res = (index="media_articles",doc_type="articles",body="query":"match":"content":"query":query

- Query 10: This will search with query

  query=input('Enter the query:')

  Output: res = (index="media_articles",doc_type="articles",body="query":"match":"content":"query":query

7 Evaluation

Mean Average Precision (MAP) is for a set of queries is the mean of the average precision scores for each query, which is a matric for measuring the accuracy of object query detected correctly. The MAP can be calculate by using below formula where Q is number of queries:

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q))}{Q} \tag{7.1}$$

We have defined mapr() function in the python using two parameters precision and recall to calculate MAP, as described in below steps:

- Initialising average precision and average recall values to 0.
- for each loop if a new document is found, precision and recall will be updated for every document
- else, precision and recall value will be same and document number keeps increasing
- function will calculate mean average precision and mean average recall and then print them at each iteration
- defined an input function index() = input('Is the file Indexed in Elastic Search?')
- if entered no/No, another input function ch() will pop up, ch=input('Would you like to query the search engine? press y/n:')
- if entered 'y' (yes), we find all the document index id, and sort the indexes, then compare these relevant indexes to all indexes in document to create a boolean model.
- the function will return mapr(precision, recall).

Figure 15 showing the mean average precision for against query 5

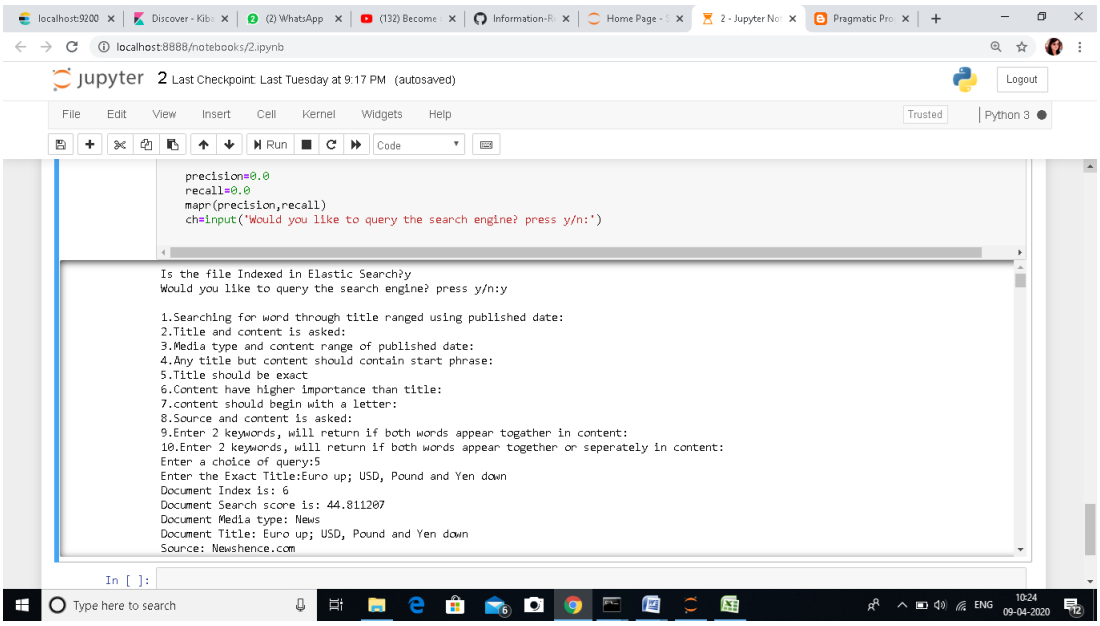


Figure 15: MAP for query 5

Figure 16 showing the mean average precision for against query 2

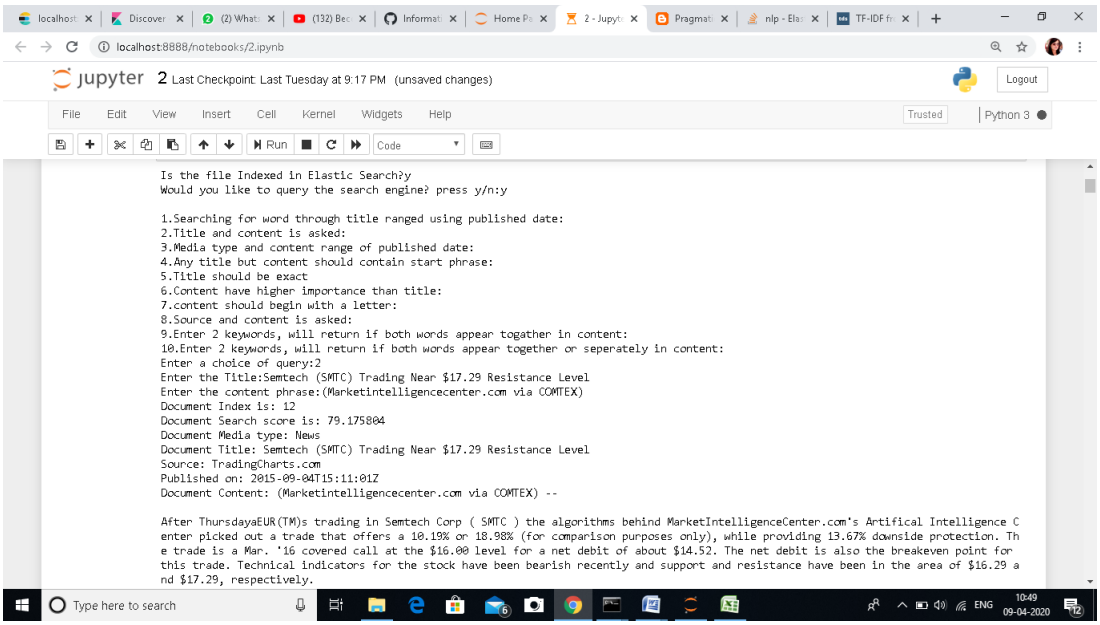


Figure 16: MAP for query 2

8 Crowdsourcing

Crowdsourcing is the practice of obtaining information or input into a task or project by enlisting the services of a large number of people, either paid or unpaid, typically via the Internet. Crowdsourcing mechanisms are now routinely being used for labelling data for information retrieval and other computational intelligence applications. We have participated in two task containing a series of 7 minutes long news videos and to recognise them. In the first task we have given with a series of 7 minutes long news videos and the task was to recognise repetition of the same, the task last for 20 minutes. We have to participate in the second task for the crowdsourcing after 24 to 72 hours post completion of the second one. The aim of the task to recognise the visuals shown in the first task. The aim of this experiment was to check visual recognition ability of human and how often we are likely to get confused seeing similar visuals.

References

[1] The Signal Media One-Million News Articles Dataset
https://research.signal-ai.com/newsir16/signal-dataset.html

[2] Web searching and information retrieval doi = 10.1109/MCSE.2004.24

[3] Wikipedia result for MAP
https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)Mean_average_precision

[4] Elasticsearch download link
https://www.elastic.co/downloads/past-releases/elasticsearch-7-6-2

[5] Kibana download link
https://www.elastic.co/downloads/kibana

[6] The Signal Media One-Million News Articles Dataset
https://research.signal-ai.com/newsir16/signal-dataset.html

[7] Kibana Indexing Tutorials
https://www.elastic.co/guide/en/kibana/current/index-patterns.html

[8] Kibana Discover Tutorials

https://www.elastic.co/guide/en/kibana/current/tutorial-discovering.html