Department of Computing and Electronic Systems, University of Essex

# Assignment 2: Elasticsearch and Evaluation

Submitted as part of the requirements for:

**CE706 – Information Retrieval**

**Students Names**:

Pranavi Vasa – 1806152

Siddharth Shyamsunder – 1802772

**Supervisor**: Kruschwitz Udo

**Date**: 22 March 2019

# 1 Required Software / Hardware

## 1.1 Software

This section explains what all the software we have used, and requirements needed to run the software. It also explains about the tools/packages we have used in this assignment.

1) **Elasticsearch:** we have used elasticsearch-6.5.4. First in order to search the software we need make sure that elasticsearch is up and running. In order to run this go to the folder in which elastic search is installed and type "bat\elasticsearch.bat" query in the console(or powershell). Figure-1 shows running of the elasticsearch.

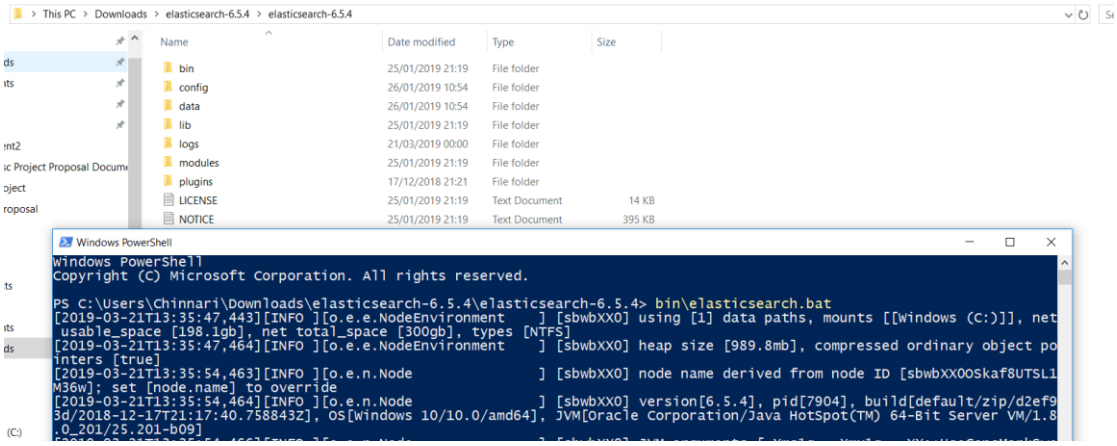


Figure-1: Running elastic search

After starting the elasticsearch go to the web browser and type "http://localhost:9200" then it will show cluster name, version and all the details as shown in Figure-2.

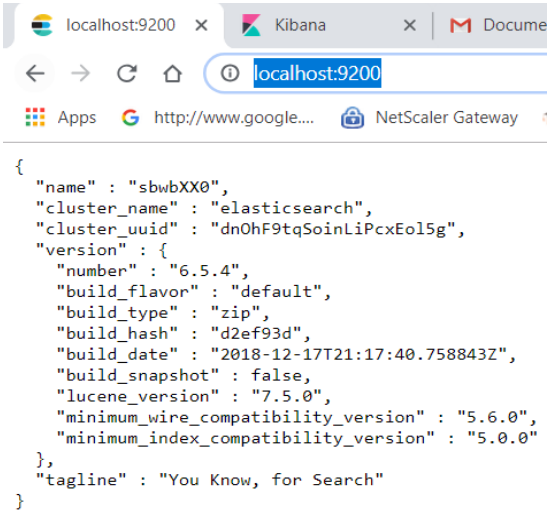


Figure – 2: Elastic search is up and running.

2) **Kibana:** For searching the index we have created we are using Kibana. For this we have used kibana-6.5.4. To the kibana go to the folder and run "bin\kibana.bat" command in powershell. Figure – 3 represents running the command. Make sure elasticsearch is up and running before starting kibana.

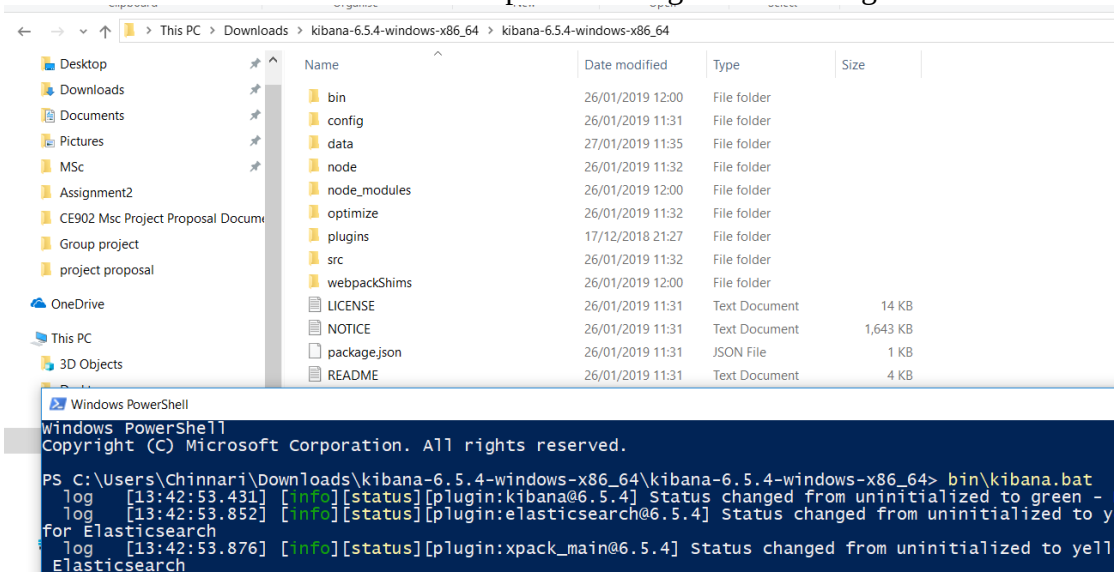


Figure – 3: Running of kibana

To check whether kibana is running or not. Type "http://localhost:5601" in web browser we are able to see the below screen. Figure – 4 shows the running state of the kibana.

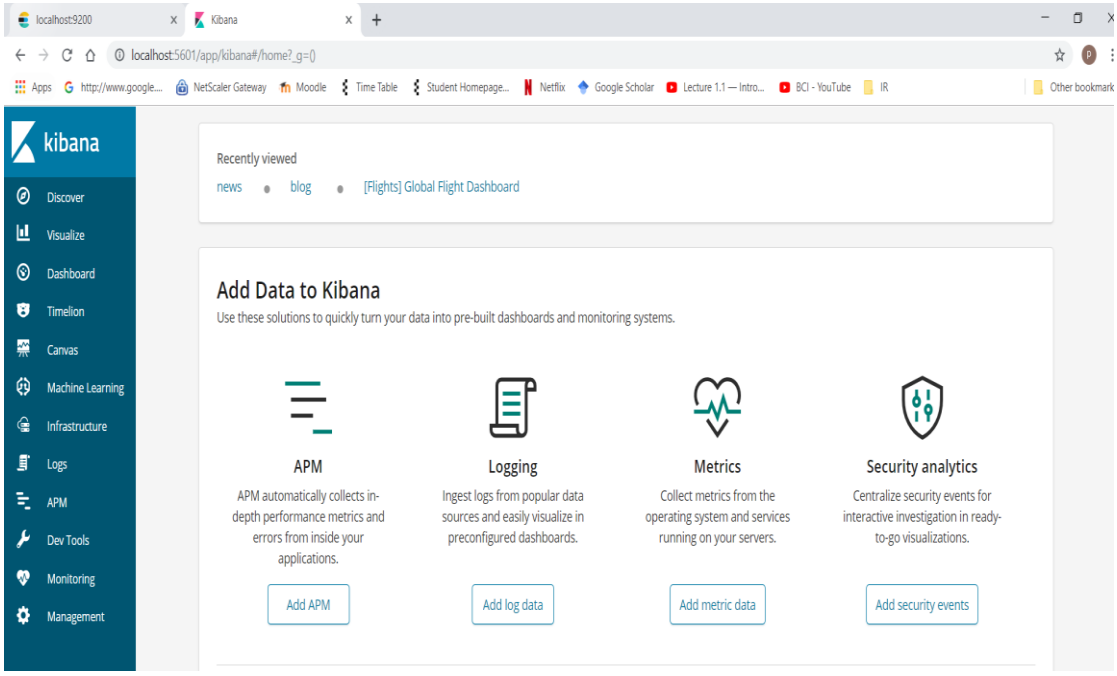


Figure – 4: Running state of kibana.

3) **Python:** We have used python 3.6.5. we are using python to load the data into kibana.

## 1.2 Packages

1) **json_lines:** json_lines package is used to read the lines
2) **json:** json package is used for converting string to json file and also for read the json file.
3) **elasticsearch:** we use elasticsearch package to connect to the elasticsearch running on the system.

We need to download all of the above packages. We can download this by using "pip install package_name" in windows.

# 2 Indexing

In this we are using 5000 articles. We have used python to index the documents. Below are the steps for indexing.

1) We have created index with the help of python.
2) First downloaded the file "sample-1M.jsonl". And opened the file.
3) jsonl(jasonlines) is nothing but a file in which each line represents a json file. We have read the file using json_lines pakage.
4) When we read the data from the file the output is in the form of string. So we have converted the data into json format using json package. Used json.dumps() to convert the data into json format.
5) After that we have read the file to load the file into elastic search.
6) In the end we have created index named "news_article" and adding the files to the index (for initial one if the index is not present then it will create the index)

All the above steps are done in indexing() function in "search_engine.py" file. We can see this index in kibana after running the file successfully. In kibana go to Management -> Index Management. There we can able to see the created index. Figure-4 and Figure-5 represents the same.

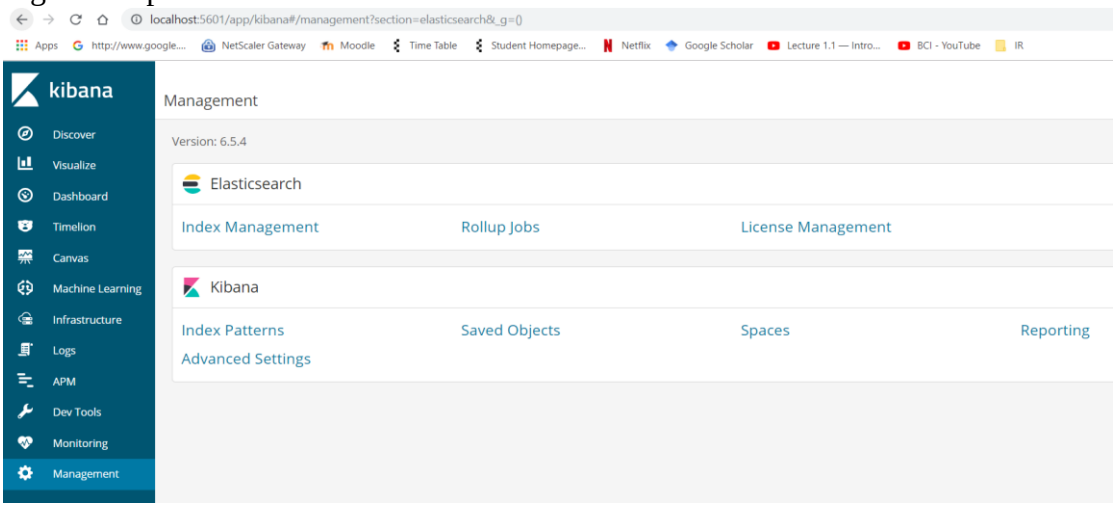


Figure – 4: Checking the index

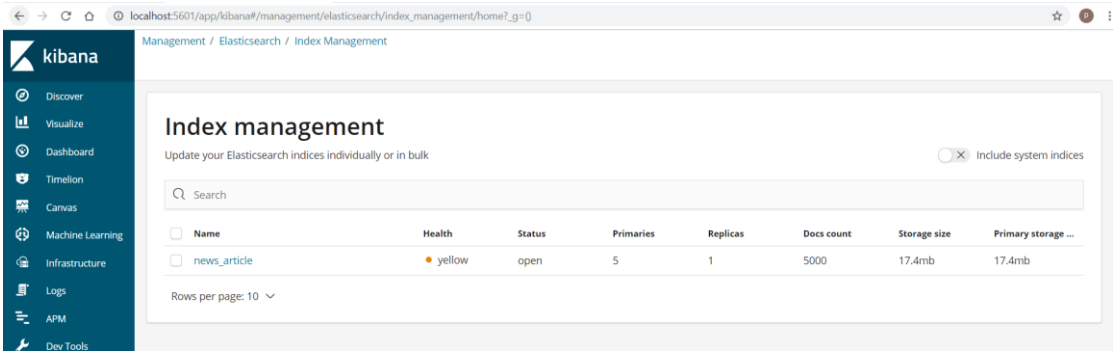


Figure-5: Checking the index

We can able to see the mappings, if we click on the index and then mappings tab. Figure-6 shows the mappings. These mappings are automatically created while we are loading the data into elasticsearch.

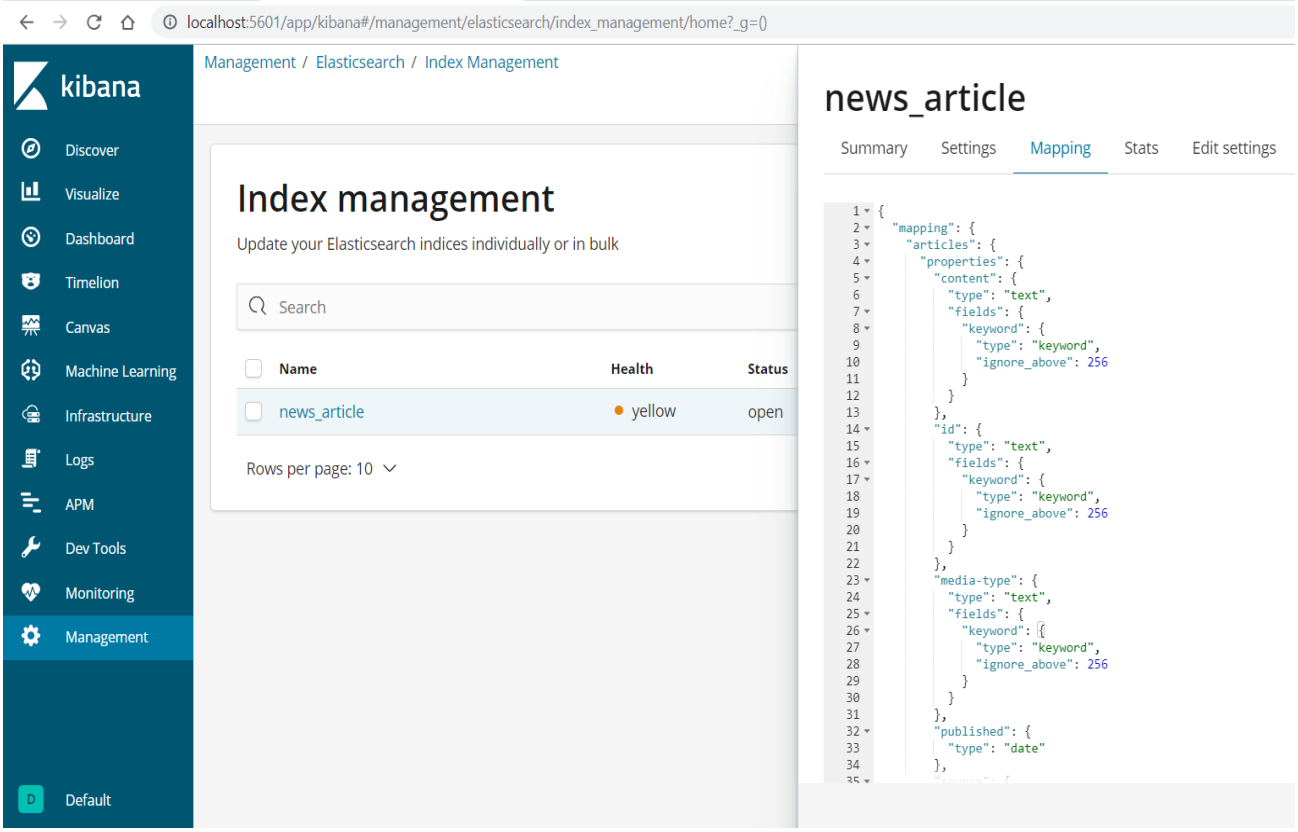


Figure-6: mappings

The loaded json data consists of 6 fields namely id, content, title, media-type, source, and published. Figure – 7 represents a json file.


```
{'_index': 'news_article', '_type': 'articles', '_id': '2', '_version': 3, 'found':
{'id': '1aa9d1b0-e6ba-4a48-ad0c-66552d896aac',
'content': 'Home »\rStyle » The Return Of The Nike Air Max Sensation Has 80's Babies
 you were lucky enough to witness the beauty that is 90s basketball. It was truly a
 wearing on their feet you would have also noticed the wide array of footwear these
 set to receive the retro treatment this year! Originally released back in …read mo
 'title': 'The Return Of The Nike Air Max Sensation Has 80's Babies Hyped!',
 'media-type': 'Blog',
 'source': 'Streets Connect',
 'published': '2015-09-22T22:54:37Z'}}
```


Figure-7: Example json file

# 3 Searching

We are able search the files uploaded with the help of the uploaded index in kibana. First we created a index pattern in management -> index patterns -> then click on create index pattern. Have created new index pattern named “news_article*”.

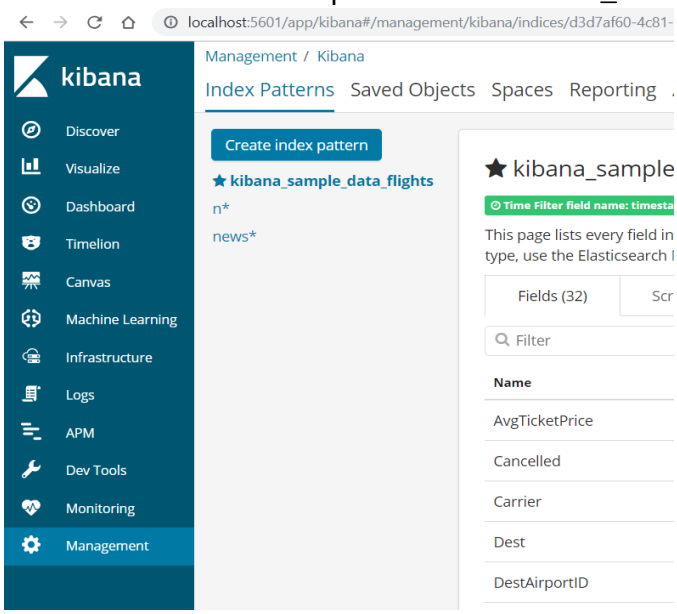


Figure-8: creating index pattern

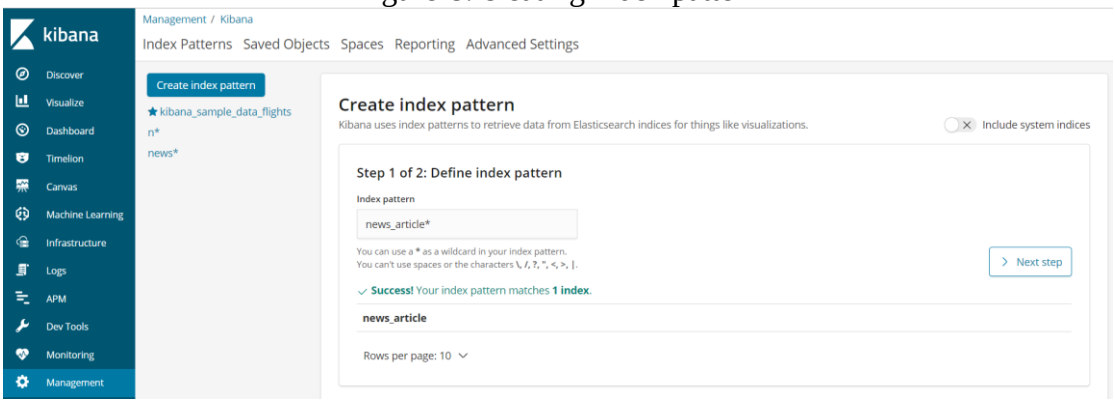


Figure-9: creating index pattern.

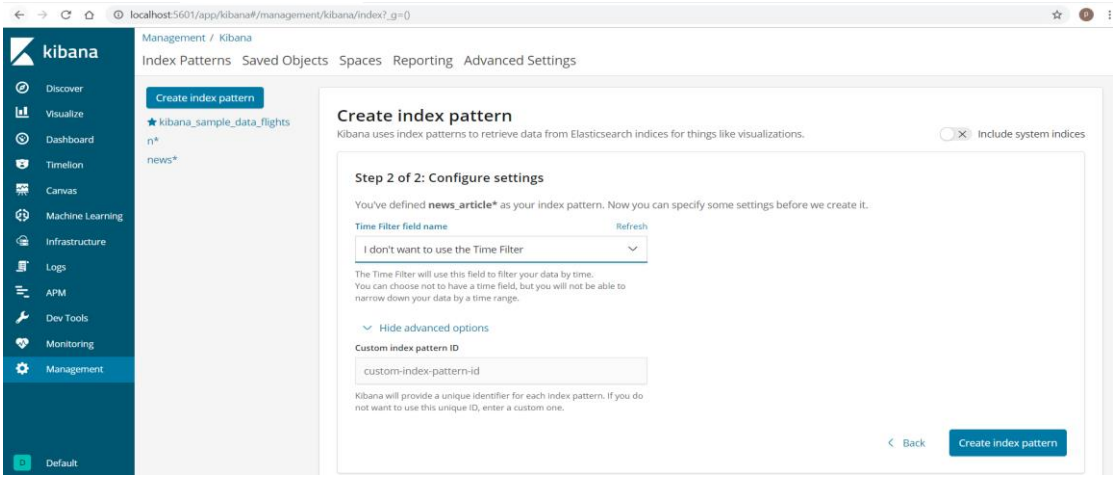


Figure – 10: Creating index pattern

Figure 8, 9 and 10 represents the creation of the index.

Once we create the index, we can be able to search in discover tab of the kibana. First change the time range value to last 5 years.

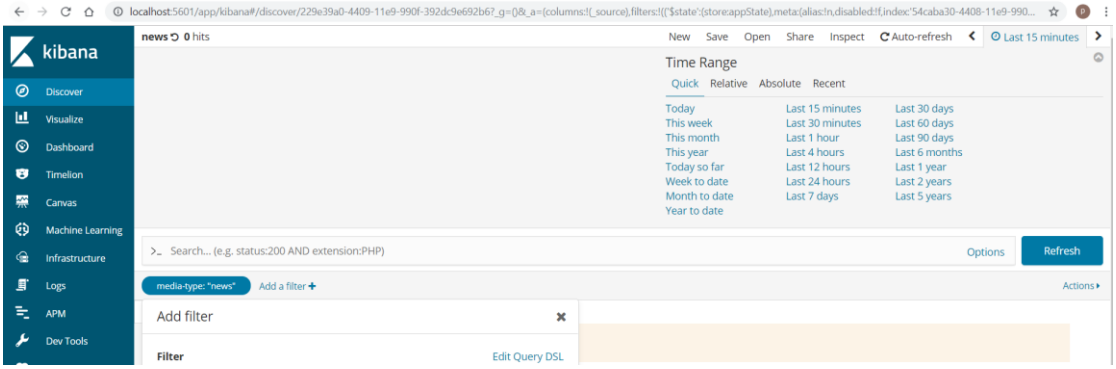


Figure – 11: Setting the time

We will give some examples of how we will search:

1) Searching for media-type is news.
   For doing this we need to add filter in kibana. Figure-12 shows how to search kibana.

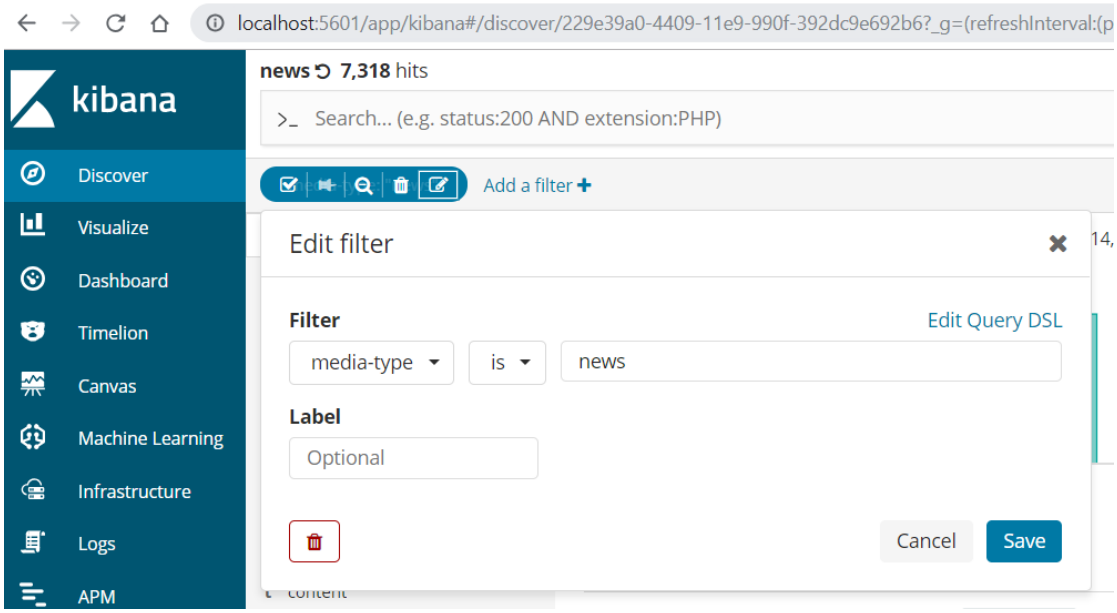


Figure – 12: Searching for media-type news.

Once we save the search we are able to see the search results. We can see all the media type as news  in Figure -13.

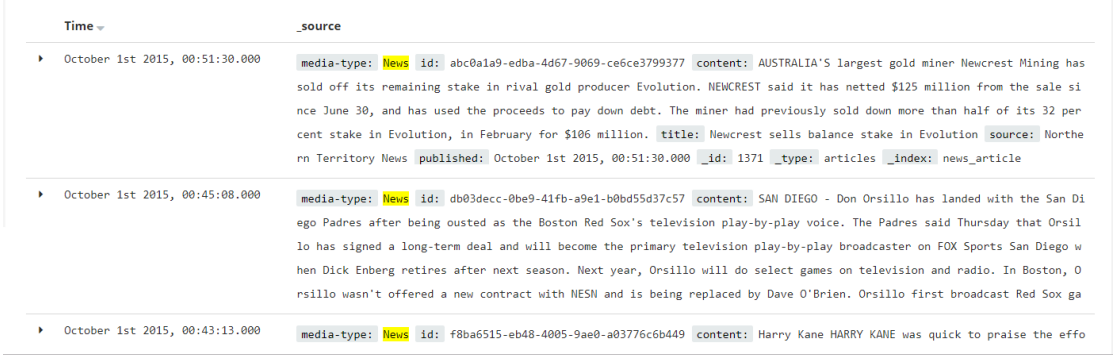


Figure – 13: output of the query 1

2) Searching with multiple filters. Add another filer saying content type as “Don*” then it will return all the articles that contains Don in it. Figure – 14: represents the outputs of the same.


Figure -14: output of the query 2

3) Searching with the media-type bolg. It shows 1360 hits. That means in the uploaded data 1360 articles contains media-type as blog.

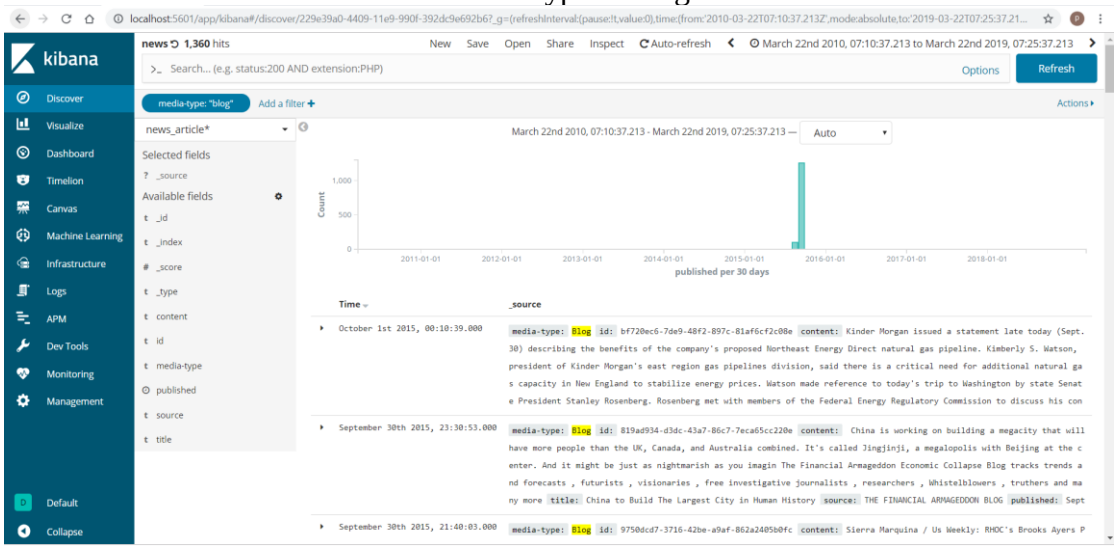


Figure-15: output for query 3.

4) This search is for title and content type. If we give search and title fields then it will show only 1 result in the output. Because it matches exact content.


Figure-16: output for query 4.

# 4 Building a test collection

We are tried to find outputs of 10 different queries.

1) Query 1: In this we will be searching for word through title ranged using published date.

Query 1:

```
{"query":
        {"bool":
                {"should":
                        [
                                {"match":
                                        {"title":title
                                        }
                                },
                                {"range":
                                        {"published":
                                                {"gte": start,"lte": end,"format":
"yyyy/MM/dd||yyyy"
                                                }
                                        }
                                }
                        ]
                }
        }
},size=5000
```

2) Query 2: this will search title and content of the article.

```
{"query":
        {"bool":
                {"should":
                        [
                                {"match":
                                        {"title":title
                                        }
                                },
                                {"match":
                                        {"content":content
                                        }
                                }
                        ]
                }
        }
},size=5000
```

3) Query 3: this will search Media type and content range of published date

```
{"query":
        {"bool":
                {"should":
                        [
                                {"match":
                                        {"media-type":media
                                        }
                                },
                                {"match":
                                        {"content":content
                                        }
                                },
                                {"range":
                                        {"published":
                                                {"gte": start,"lte": end,"format":
"yyyy/MM/dd||yyyy"
                                                }
                                        }
                                }
                        ]
                }
        }
},size=5000
```

4) Query 4:

```
{"query":
        {"bool":
                {"should":
                        [
                                {"match":
                                        {"title":title
                                        }
                                },
                                {"match_phrase_prefix":
                                        {"content":content
                                        }
                                }
                        ]
                }
        }
},size=5000
```

5) Query 5:this matches with the exact title content type

```
{"query":
        {"match_phrase":
                {"title":title
                }
        }
},size=5000
```

6) Query 6: Content have higher importance than title:

```
{"query":
        {"multi_match":
                {"query":keyword,"fields":
                        ["content^2", "title"
                        ]
                }
        }
},size=5000
```

7) Query 7: content should begin with a letter:

```
{"query":
        {"wildcard":
                {"content": content
                }
        }
},size=5000
```

8) Query 8: Source and content is asked

```
{"query":
        {"bool":
                {"should":
                        [
                                {"match":
                                        {"source":source
                                        }
                                },
                                {"match":
                                        {"content":content
                                        }
                                }
                        ]
                }
        }
},size=5000
```

9) Query 9: if you enter 2 keywords, will return if both words appear together in content

```
{"query":
        {"match":
                {"content":
                        {"query":query,"operator":"and"
                        }
                }
        }
},size=5000
```

10) Query 10: if we enter 2 keywords, will return if both words appear to together or seperately in content

```
{"query":
        {"match":
                {"content":
                        {"query":query,"operator":"or"
                        }
                }
        }
},size=5000
```

# 5 Evaluation

We are calculating Mean average precision (MAP) here. It is calculated in mapr() function.

1) First, we will initialise precision and recall values to 0.
2) Here we will calculate ranked recall and precision for the search engine.
3) For each of the document retrieved it will store the all index values in an array and sort the array. This will help as the while calculating precision and recall values
4) Precision and recall will be calculated at every stage:
    a. This can be explained with the help of an example:
    b. suppose we have 10 documents in the index, if we query the index then it will return 3 documents.
    c. Id's of the documents returned are 2,4,7 then precision and recall calculated at the each step means if we take 1st document then it will not return anything.
    d. So precision will be 0/1 and recall will 0/10. In 2nd stage it is returning 2nd document so precision is ½ and recall 1/10. Like this the process goes on.
5) Once we calculate the precision and recall then we will try to average all values of precision and recall.
6) Figure – 17 gives the output of the this function. It shows id of the document, precision, recall at the end it will show average of the precision and recall.

```
4981 0.0002007628990162618 0.0002 3
4982 0.0002007226013649137 0.0002 3
4983 0.0002006823198876179 0.0002 3
4984 0.0002006420545746388 4 0.0002 3
4985 0.0002006018054162487 4 0.0002 3
4986 0.0002005615724027276 3 0.0002 3
4987 0.0002005213555243633 4 0.0002 3
4988 0.0002004811547714514 8 0.0002 3
4989 0.0002004409701342954 6 0.0002 3
4990 0.000200400801603206 4 0.0002 3
4991 0.0002003606491685033 0.0002 3
4992 0.0002003205128205128 1 0.0002 3
4993 0.0002002803925495694 0.0002 3
4994 0.0002002402883460152 2 0.0002 3
4995 0.0002002002002002002 0.0002 3
4996 0.0002001601281024819 7 0.0002 3
4997 0.0002001200720432259 3 0.0002 3
4998 0.0002000800320128051 4 0.0002 3
4999 0.0002000400080016003 2 0.0002 3
5000 0.0002 0.0002 3
5001 0.0001999600079984003 0.0002 3
Mean Average Precision: 0.0014522750959318213
Mean Average Recall: 0.0001999199999998452
```

Figure – 17: output

# 6 Complete system

- Our system integrates python with elastic search(search_engine.py). So first it will ask the whether the articles are indexed or not. If not then it will index the collection.
- After that it will display list of options to the users from which user will select one option.
- Then it will query elasticsearch and output the results.
- After that it will calculate the precision and recall at every stage of the project.
- Finally it will give the mean average precission.
- Figure – 18 and 19 shows how the system works. Output of the system for one of the queries is attached.

```
Is the file Indexed in Elastic Search?y
Would you like to query the search engine? press y/n:y

1.Searching for word through title ranged using published date:
2.Title and content is asked:
3.Media type and content range of published date:
4.Any title but content should contain start phrase:
5.Title should be exact
6.Content have higher importance than title:
7.content should begin with a letter:
8.Source and content is asked:
9.Enter 2 keywords, will return if both words appear togather in content:
10.Enter 2 keywords, will return if both words appear together or seperately in content:
Enter a choice of query:5
Enter the Exact Title:The Return Of The Nike Air Max Sensation Has 80's Babies Hyped!
Document Index is: 3
Document Search score is: 55.381924
Document Media type: Blog
Document Title: The Return Of The Nike Air Max Sensation Has 80's Babies Hyped!
Source: Streets Connect
Published on: 2015-09-22T22:54:37Z
Document Content: Home »Style » The Return Of The Nike Air Max Sensation Has 80's Babies Hyped
enough to witness the beauty that is 90s basketball. It was truly a great time to be basketbal
have also noticed the wide array of footwear these player used to rock. One of those happens t
Originally released back in …read more Author: KicksOnFire      Share This Post OnGoogleFacebo

2 0.0 0.0 3
3 0.0 0.0 3
4 0.25 0.0002 3
5 0.2 0.0002 3
6 0.16666666666666666 0.0002 3
7 0.14285714285714285 0.0002 3
8 0.125 0.0002 3
9 0.1111111111111111 0.0002 3
10 0.1 0.0002 3
11 0.09090909090909091 0.0002 3
12 0.08333333333333333 0.0002 3
13 0.07692307692307693 0.0002 3
```

Figure – 18: Entire search engine.

```
4994 0.00020024028834601522 0.0002 3
4995 0.0002002002002002002 0.0002 3
4996 0.00020016012810248197 0.0002 3
4997 0.00020012007204322593 0.0002 3
4998 0.00020008003201280514 0.0002 3
4999 0.00020004000800160032 0.0002 3
5000 0.0002 0.0002 3
5001 0.00019996000799840003 0.0002 3
Mean Average Precision: 0.0014522750959318213
Mean Average Recall: 0.00019991999999998452
Would you like to query the search engine? press y/n:
```

Figure – 19: Entire search engine.

Further other pre-processing steps can be applied to the data and again it can be loaded to the elasticsearch.