**CE706 - Information Retrieval SU 2022**

Assignment- 1

*Name*: Renjith Nataraja Pillai

*Student ID:*2111151

# Instructions For Running The System

Elasticsearch running in windows operating system is used for completing the assignment. Kibana which acts as a Elasticsearch web interface is used to visualize the data ,it also allows to visualize the query results for better analysis. Even though the REST API is enough to perform all the tasks, python library for elastic search is also used to manage Elasticsearch by writing python code.

In order to run the system first download Elastic search and Kibana from the official website and install Elasticsearch library in python API.

Step 1:

* From the official website of the elasticsearch, Elasticsearch and Kibana zip files for windows are downloaded and unzipped into a folder named ElasticStack .

Step 2:

* After downloading and unzipping ,first we have to install Elasticsearch into the system.
* Go to the extracted elasticsearch folder select the folder ‘config’

open the file ‘elasticsearch.yml’ in text editor.

* Next step is to give a cluster name and node name .The cluster is named as IR\_asignment and node as Signal\_media

Text

Description automatically generated

Fig:1

* In order to disable the security write the code **‘xpack.security.enabled: false’**

at the end . This will prevent sniffing the token from a connection over plain **http.xpack.**

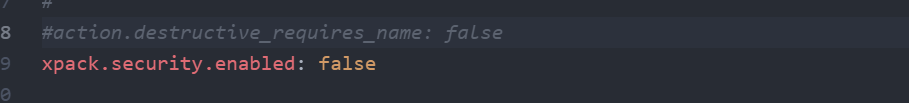


Fig:2

* Now going back to the Elasticsearch folder again and select the ‘bin’ folder. Copy the path of the same which will use to establish the connection with Elasticsearch server.
* Open the command prompt and select run as administrator. In the terminal opened, paste the file path of the bin folder followed by ‘cd’ and press ‘Enter’.it successfully changed into the bin directory.
* Write the command ‘elasticsearch.bat’ in the terminal ,which will run the Elasticsearch server on local host at port 9200.

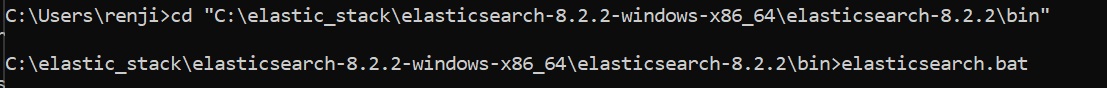


Fig:3

* In order to check whether the connection is established or not ,open the web browser and go to the link <http://localhost:9200>. If the connection is successful, the following message will show up.

Text, letter

Description automatically generated

Fig:4

Step:3

* After connecting to elasticsearch server ,next step is to install kibana. The procedure is same as the Elasticsearch, except we don’t need to change anything in config.
* In the Kibana folder copy the ‘bin’ folder path
* Open the command prompt and select run as administrator.In the terminal opened, paste the file path of the bin folder followed by ‘cd’ and press ‘Enter’.It successfully changed into the bin directory.
* Write the command ‘Kibana.bat’ in the terminal ,which will run the elasticsearch server on local host at port 5601

Graphical user interface, text

Description automatically generated

Fig:5

* In order to open the Kibana interface, go to <http://localhost:5601> .
* For writing the queries in Elasticsearch click on the ‘menu’ option and select ‘Dev tools’.

The queries can be written in the left side of the console and results will show up in the right side.

Graphical user interface, text, application

Description automatically generated

Fig:6

Step:4

* Finally install the Elasticsearch library in Python ,using the command **pip install elasticsearch**
* For the convenience jupyter notebook IDE is used. In order to connect Elasticsearch to python run the command:

A picture containing text

Description automatically generated

Fig:7

* Check whether the connection is established or not by writing the function:

Graphical user interface, text, application

Description automatically generated with medium confidence

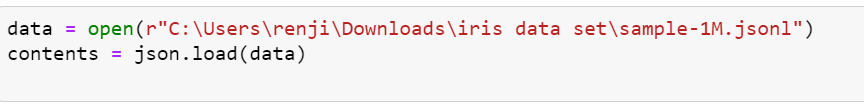
Fig:8

# Indexing

An index can be defined as a collection of documents or a database and the process indexing is a way to optimise the database performance. The task here is to index the Signal Media One Million News Article dataset,in order to do that first we have to download the dataset.

* The dataset can be downloaded using the link ***https://research.signal-ai.com/newsir16/signal-dataset.html*** . The dataset is a compressed jsonl file in the format ‘jsonl.gz’.
* For the convenience the compressed dataset is unzipped into jsonl file using winzip. Since we are using python for indexing the dataset ,jsonl file works well in python.
* Attempting to upload and index the whole file in a single run.
* The next step is to upload the file in python. Json library is used to upload the data using load() function.

Issue faced while loading:



Code. Fig:9

Graphical user interface, text, application

Description automatically generated

Error message. Fig:10

While running the code an error **‘Extra data: line 2 column 1 (char 2598)’** occurred. This is actually a decode error which will occurs due to the format of the json file that we are trying to upload. This may happen when the Json file contains multiple strings. The **Jason.load()** function can parse only one string at a time. As a solution to that we have to upload the data line by line.

A for loop is used to iterate over the data and the data is upload at each iteration and stored it in a dictionary.

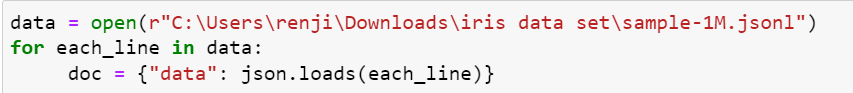


Fig:11

* The second part is Indexing, for that Elasticsearch’s **index()** function is used. The index function is given inside the For loop so that the indexing will happen at each iteration.

Issues faced while indexing:

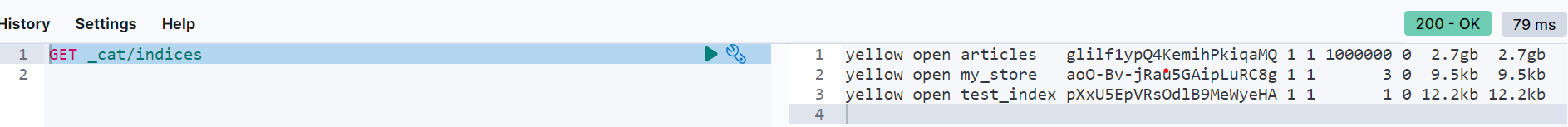
Since the file is large with around 1 million data, it will take much time for indexing. In the first attempt the connection interrupted due to network issues. The partially uploaded index was deleted and indexing is started from the beginning.it took around two hours to index the entire document and it was successful.

Graphical user interface, text, application, email

Description automatically generated

Python code. Fig:12

* To confirm the indexing , run the code **GET \_cat/indices** in Kibana Dev Tool.



Timeline

Description automatically generated with medium confidence

Fig:13

An index named **“Articles”** is create with one million files in it.

* For mapping ,body is specified as input jsonl data.So it will automatically fetch the mapping and will create the fields. The fields here created are “id”, "media-type", "source", "title" and "content”. Fields types are not specified here since the default field type is text and all fields are in text format.

Timeline

Description automatically generated with medium confidence

Fig:14

﻿Tokenization and Normalisation

An analyser in Elasticsearch defines how the text is indexed and searched. It will break the search text into individual words based on certain conditions. There are different types of analyzer’s in Elasticsearch, also it is possible to custom our own analyzer and a custom analyser is used for this assignment. There are mainly 3 components in an analyzer, one is tokenizer and the second one is token filter and the third one is called character filters.

Tokenizer splits the text into individual tokens ,based on certain conditions. There are different types of tokenizers are available in Elasticsearch .(eg: whitespace, ngram etc).Token filters receive the tokens and add some filter to the tokens ,like removing the stop words, changing the characters from upper case to lower case etc.

In order to apply the custom analyzer ,the index settings should be updated. For that a command PUT articles/\_settings is used. The tokenizer used here is “white space tokenizer” , with “stop” and “lowercase” filters .The white space tokenizer will split the text into terms whenever it encounters a whitespace character and the stop filter will remove the stop words from the tokens such as : “the”, “there”, ”an” etc.The lowercase filter will convert all letters into lower case.

While setting up the tokenizer, the below error occurred:

Graphical user interface, text, application

Description automatically generated

Fig:15

This error occurred because the index “articles” is still open .it is not possible to update an index settings when its open. In order to close the index, the command POST /articles/\_close is used.

Graphical user interface, application, Word

Description automatically generated

Fig:16

Sample text:

Graphical user interface, application, Teams

Description automatically generated

Fig:17

In the above example the analyzer will convert all the letters into lowercase .

Consider an another case: what if the search text has some symbolic Unicode characters? This time the analyzer may not find the texts corresponding to the Unicode.so we have to convert them into ASCII equivalent. “ASCIIFoldingFilter” can be used for that purpose.

Graphical user interface, text, application

Description automatically generated

Fig:18

In the above example the filter changes à to a. But what if the text have non-alphanumeric characters like “+”, “-“ etc. Lets try an example

*Graphical user interface, text, application

Description automatically generated*

Fig:19

It is clear that the analyzer was not able to separate the special characters in the text. In order to do that we have to add a token filter called "word\_delimiter". This filter will split the tokens at special characters.

*Graphical user interface, text, application

Description automatically generated*

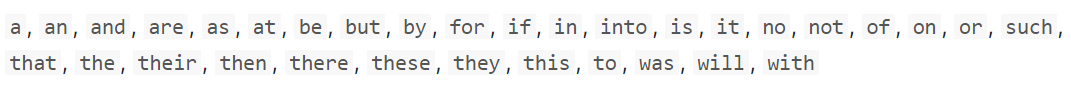
Fig:20

﻿ Now the tokens are perfectly splitted at spaces and special characters.

Selecting Keywords

The next task is to identify the words that are most useful for indexing or in other words ,removing the words that are not useful for indexing. So its must be specified to the analyser that these words must be ignored while searching as well as in the retrieving phase. These words are called stop words. Mostly they are the frequently occurring words(eg: “the”, ”was”, ”in” etc…) that are irrelevant for our Searching purpose.

In elastic search there is a build in token filter called “stop” to remove these irrelevant words. It will remove the following English stop words by default:



Consider the above example , in the sample string "The IÀNS India-Private+ Limited" ,we need to remove the stop word “the”. For that the filter “stop” must be added to the analyzer.

Graphical user interface, text, application

Description automatically generated

Fig:21

While retrieving, the stop word “the” is not considered.

**N-gram Tokenizer:**

The N-gram tokenizer first breakdown the text into words and the words are again split into N\_grams based on the specified length . consider an sample text:

***"The IÀNS India-Private+ Limited"***Now applying N- gram tokenizer:

Graphical user interface, text, application

Description automatically generated

Fig:22

Here the the N gram tokenizer consider the initial text as a single token and create n grams with maximum and minimum length of three. N grams that are configured like these are called tri grams.

**TF-IDF Score**

When the user passes a search query ,the system should identify how relevant that word to the entire document. There are statistical measures to identify it. Most commonly used method in elastic search is the TF-IDF Score ,it will score a word by multiplying the term frequency(TF) with Inverse term frequency(IDF). In elastic search the similarity module can be used to find out the TF-IDF Score.

Lets try an example:

Graphical user interface, application, Teams

Description automatically generated

Fig:23

The similarity used here is “BM25” which is a TF/IDF based similarity that has built-in tf normalization. Default values are given to the parameters “b” and “k1”. Word relevance is more when the score is high.It will also display the maximum score of that particular word in the whole document.

# Stemming or Morphological Analysis

Stemming is the method of reducing a word into its root form.This will hleps to get better results during search. In Elasticsearch the stemming is done using token filters. Lets consider a sample text:

***"the oranges are ripening fast"***

While stemming ,the filter will convert the words “oranges” and “ripening” into its root form “orange” and “ripen” respectively. Stemming filter used here is “kstem” which combines algorithmic stemming with a built-in dictionary and added it to the analyzer:”my\_analyzer2”

Graphical user interface, application

Description automatically generated

Fig:24

# Searching

Lets try some Search queries that user will come up:

**1.Find the articles published in a particular date:**

If we want to find the articles that publishes in a particular date ,we can use Range query on date fields. Here the date field is “published”:

Graphical user interface, text, application

Description automatically generated

Fig:25

In this example,articles that are published on the date "2015-09-05" is filtered.

**2.Find the articles published in a specific date and source.**

In the above example only date is filtered ,what if the user wants to find the articles published by a specific source and in a particular date. In order to do that compound queries should be used. The most commonly used is “Bool Query” which is a default query for combining multiple queries.

Graphical user interface, application

Description automatically generated

Fig:26

Here Bool query is used to search an article published in the date "2015-09-01" and by the source “yahoo sports”

**3.Find the articles related to a particular content and published in a specific date and source.**

What if there are more than two queries for a single search?.For example the user needs to search an article article related to a particular content and published in a specific date and source.In those cases a single must clause with a multiple bool queries inside should use for search.

Graphical user interface, text, application

Description automatically generated

Fig:27

In this example the user is searching an article related to “ Randy Moss boot camp” that published in the date "2015-09-28" and by the source “yahoo sports”.