CE706 - Information Retrieval 2021

Assigment 1

Registration Number : 2004351

1) Instructions for running your system (Engineering a Complete System) :

- *There are following steps one should follow to perform all operations on elastic search :*

1. *Install Python Programming Language with latest stable compiler 3.8.8*
2. *The Project has some module dependencies like pandas, nltk,. I installed all the module dependencies to fulfil the project requirements.*

```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\user\miniconda3\envs\ce802\lib\site-packages (1.1.3)
Requirement already satisfied: pytz>=2017.2 in c:\users\user\miniconda3\envs\ce802\lib\site-packages (from pandas) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\user\miniconda3\envs\ce802\lib\site-packages (from pandas)
(2.8.1)
Requirement already satisfied: numpy>=1.15.4 in c:\users\user\miniconda3\envs\ce802\lib\site-packages (from pandas) (1.19.1)
Requirement already satisfied: six>=1.5 in c:\users\user\miniconda3\envs\ce802\lib\site-packages (from python-dateutil>=2.7.3->
pandas) (1.15.0)
```

```
pip install nltk
```

```
Collecting nltk
  Downloading nltk-3.5.zip (1.4 MB)
Requirement already satisfied: click in c:\users\user\miniconda3\envs\ce802\lib\site-packages (from nltk) (7.1.2)
Requirement already satisfied: joblib in c:\users\user\miniconda3\envs\ce802\lib\site-packages (from nltk) (0.17.0)
Collecting regex
  Downloading regex-2020.11.13-cp37-cp37m-win_amd64.whl (269 kB)
Collecting tqdm
  Downloading tqdm-4.58.0-py2.py3-none-any.whl (73 kB)
Building wheels for collected packages: nltk
  Building wheel for nltk (setup.py): started
  Building wheel for nltk (setup.py): finished with status 'done'
  Created wheel for nltk: filename=nltk-3.5-py3-none-any.whl size=1434679 sha256=7e82f75cbeaa39da5c3d9ba7359c127712b455aedaf8ad
0692263f5240528e49
  Stored in directory: c:\users\user\appdata\local\pip\cache\wheels\45\6c\46\a1865e7ba706b3817f5d1b2ff7ce8996aabdd0d03d47ba0266
Successfully built nltk
Installing collected packages: regex, tqdm, nltk
Successfully installed nltk-3.5 regex-2020.11.13 tqdm-4.58.0
```

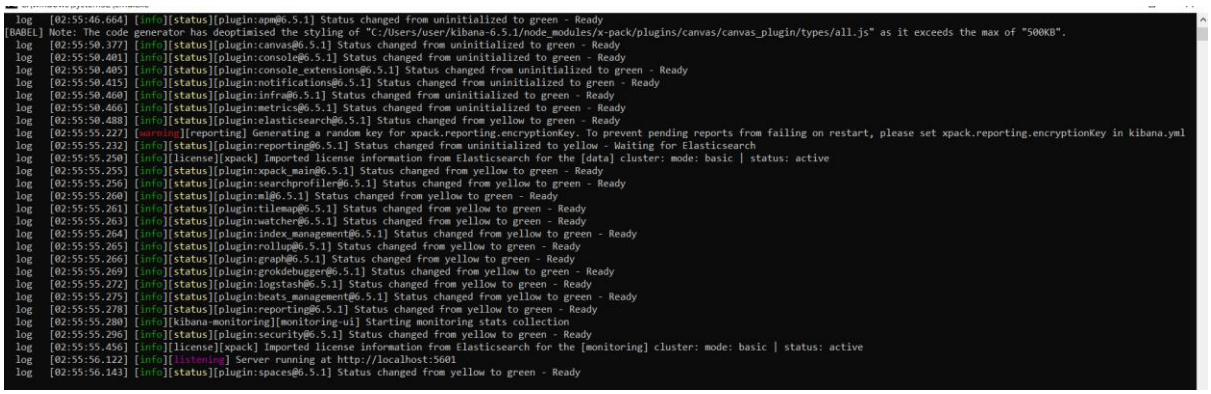
3. *Install Elastic Search Engine for indexing and searching purpose.*

```
pip install elasticsearch
```

```
Requirement already satisfied: elasticsearch in c:\users\user\miniconda3\envs\ce802\lib\site-packages (7.11.0)Note: you may nee
d to restart the kernel to use updated packages.
Requirement already satisfied: certifi in c:\users\user\miniconda3\envs\ce802\lib\site-packages (from elasticsearch) (2020.6.2
0)
Requirement already satisfied: urllib3<2,>=1.21.1 in c:\users\user\miniconda3\envs\ce802\lib\site-packages (from elasticsearch)
(1.25.11)
```

4. *To Run Elastic Search Server download elasticsearch-6.5.1 unzip it and go to bin folder then run **elasticsearch.bat** file to run the server. The server will start on port **9200**.*

5. *I used Kibana for visualization purpose. I created index and searched the documents in command prompt as well as in Kibana.*
6. *So, to run the server of Kibana download the kibana-6.5.1 unzip it and go to its bin folder to run* ***kibana.bat*** *file. The Kibana server will run on port* ***5601****.*



- *After performing above all the steps one can able to solve the assignments problem steps.*

2) Indexing:

To get the Dataset:

- To get the dataset go to this described website https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge and download it from Kaggle website.
- As mentioned in the assignment I only used first 1000 documents from the metadata.csv file.
- I made one new file **covid.csv** file which is contain only first 1000 documents from metadata.csv file and load the data into frame using pandas.

```
# Read csv file into a pandas dataframe
df = pd.read_csv(r'D:\Information_Retrival\covid.csv')
print("Schema :", df.dtypes)
print("No of docs and columns :", df.shape)
```

```
Schema : Unnamed: 0          object
sha                  object
source_x             object
title                object
doi                  object
pmcid                object
pubmed_id             int64
license              object
abstract             object
publish_time         object
authors              object
journal              object
mag_id              float64
who_covidence_id    float64
arxiv_id            float64
pdf_json_files       object
pmc_json_files       object
url                  object
s2_id               float64
dtype: object
No of docs and columns : (1000, 19)
```

- Index is a mechanism that allows a user to divide the data in a different way. Elastic Search is a mechanism which uses to distribute data around the cluster.
- To create index in elastic search first connect python with the elastic search. By default port of elastic search server is 9200.

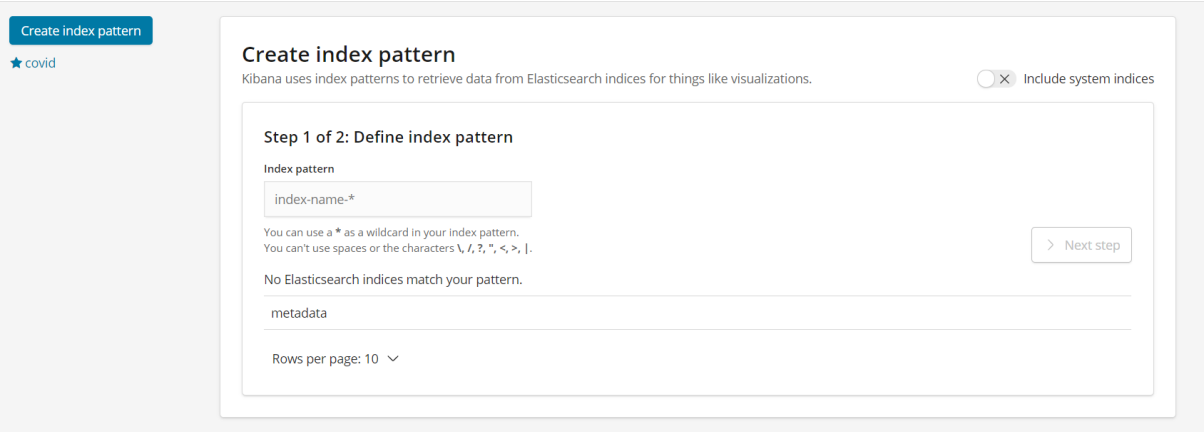
- Here, I created simple index without mapping fields. I have mentioned index name is "**covid**" and body is empty.

```
# Connection to ElasticSearch
es = Elasticsearch( ["localhost:9200"],
                    sniff_on_start=True,
                    sniff_on_connection_fail=True,
                    sniffer_timeout=60
                   )

# Simple index creation with no particular mapping
es.indices.create(index='covid',body={})
```

- It will create one index named covid in elastic search.

```
[2021-03-05T02:53:56,907][INFO ][o.e.c.r.a.AllocationService] [b4dpjpS] Cluster health status changed from [RED] to [YELLOW] (reason: [shards started [[metadata][2]] ...]).
[2021-03-05T02:55:46,836][INFO ][o.e.c.m.MetaDataIndexTemplateService] [b4dpjpS] adding template [.management-beats] for index patterns [.management-beats]
[2021-03-05T04:46:46,549][INFO ][o.e.c.m.MetaDataDeleteIndexService] [b4dpjpS] [covid/E5S-1sIjTv260lGSyWEJ_g] deleting index
[2021-03-05T04:47:07,478][INFO ][o.e.c.m.MetaDataCreateIndexService] [b4dpjpS] [covid] creating index, cause [api], templates [], shards [5]/[1], mappings []
```

- To see the created index in Kibana platform we need to go to **localhost:5601** and go to **Management > Create index pattern.**



Problems while creating an index:

- While creating an index I faced one issue which said that "**Resource is already exists**" it means the same name of index is already given earlier, that is why I need to put other name to create an index.

```
~\miniconda3\envs\ce802\lib\site-packages\elasticsearch\connection\base.py in _raise_error(self, status_code, raw_data)
    321
    322         raise HTTP_EXCEPTIONS.get(status_code, TransportError)(
--> 323             status_code, error_message, additional_info
    324         )
    325

RequestError: RequestError(400, 'resource_already_exists_exception', 'index [covid/E5S-1sIjTv260lGSyWEJ_g] already exists')
```

- To solve above issue, I have renamed the index and then tried to make new index, so It was working fine with the new name.

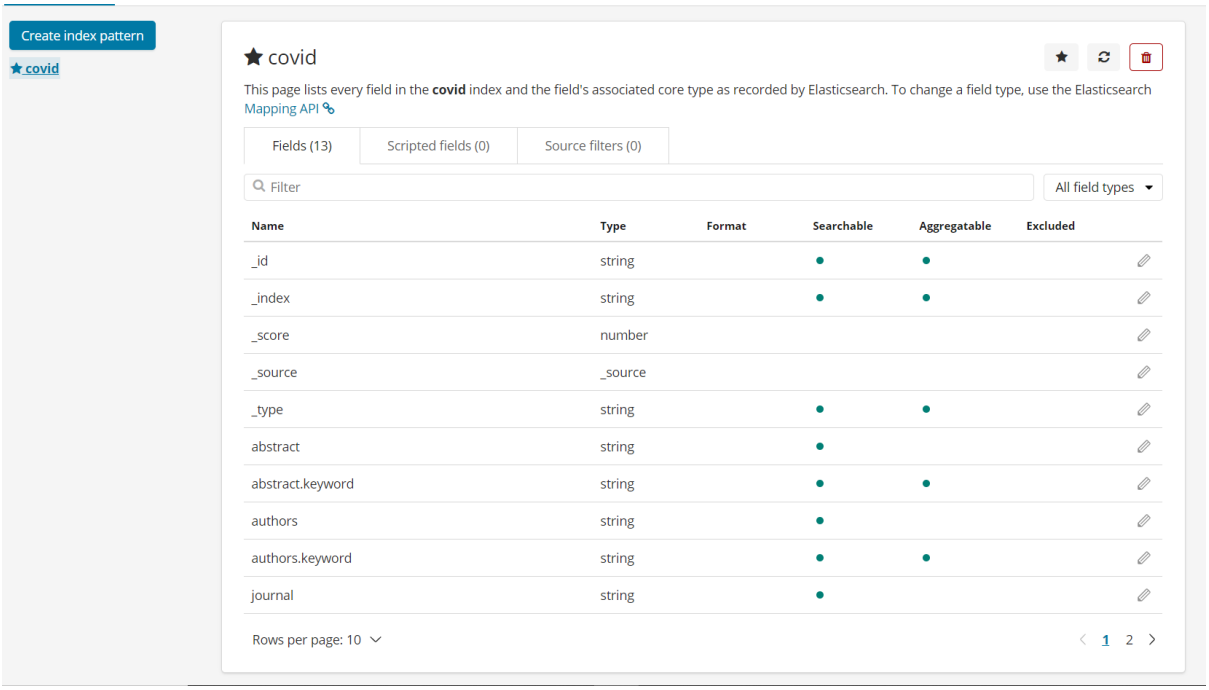
Experiments with the documents:

- I have selected major four columns from the dataset to create documents, as they have rich content. Like **"title", "abstract", "authors"** and **"journal".**

- To insert the data we need to do mapping of the columns from our data frame columns as below in code.

```
# ====== Inserting Documents ====== #

# Creating a simple Pandas DataFrame
dataframe = pd.DataFrame(data = {'title' : d["title1"], 'abstract': d["abstract1"], 'authors':d['authors1'], 'journal': d['journa

# Bulk inserting documents. Each row in the DataFrame will be a document in ElasticSearch
documents = dataframe.to_dict(orient='records')
bulk(es, documents, index='covid',doc_type='covid_data', raise_on_error=True)
```

- It will be mapping all the data with the elastic search columns name. And it will be appeared like following in Kibana visualization tool.



3) Sentence Splitting, Tokenization and Normalization:

Sentence Splitting:

- In this splitting the whole paragraph will be split into different sentences. To use Sentences, tokenize import following package from nltk.

from nltk. tokenize import sent_tokenize

- I have applied this sentence tokenize to **'title'** and **'abstract'** columns.

- Consider the example of '**title**' column.

```
from nltk.tokenize import sent_tokenize
for sentences in d['title']:
    all_sent = sent_tokenize(sentences)
    print(all_sent)
```

```
['Clinical features of culture-proven Mycoplasma pneumoniae infections at King Abdulaziz University Hospital, Jeddah, Saudi A
rabia']
['Nitric oxide: a pro-inflammatory mediator in lung disease?']
['Surfactant protein-D and pulmonary host defense']
['Role of endothelin-1 in lung disease']
['Gene expression in epithelial cells in response to pneumovirus infection']
['Sequence requirements for RNA strand transfer during nidovirus discontinuous subgenomic RNA synthesis']
['Debate: Transfusing to normal haemoglobin levels will not improve outcome']
['The 21st International Symposium on Intensive Care and Emergency Medicine, Brussels, Belgium, 20-23 March 2001']
['Heme oxygenase-1 and carbon monoxide in pulmonary medicine']
['Technical Description of RODS: A Real-time Public Health Surveillance System']
['Conservation of polyamine regulation by translational frameshifting from yeast to mammals']
['Heterogeneous nuclear ribonucleoprotein A1 regulates RNA synthesis of a cytoplasmic virus']
["A Method to Identify p62's UBA Domain Interacting Proteins"]
['Vaccinia virus infection disrupts microtubule organization and centrosome function']
['Multi-faceted, multi-versatile microarray: simultaneous detection of many viruses and their expression profiles']
['Herpes simplex virus type 1 and normal protein permeability in the lungs of critically ill patients: a case for low pathoge
nicity?']
['Logistics of community smallpox control through contact tracing and ring vaccination: a stochastic network model']
```

Word Splitting:

- In this splitting the whole sentence will be split into different words. To use Word Tokenize import following package from nltk.

from nltk. tokenize import word_tokenize

- I have applied this word tokenize to **'title', 'abstract' , 'authors'** and **'journal'** columns.
- Consider the example of '**title**' column.

```
from nltk.tokenize import word_tokenize

# Get words form 'Title' Column
for words in d['title']:
    all_words = word_tokenize(words)
    print(all_words)
```

```
['Clinical', 'features', 'of', 'culture-proven', 'Mycoplasma', 'pneumoniae', 'infections', 'at', 'King', 'Abdulaziz', 'Univer
sity', 'Hospital', ',', 'Jeddah', ',', 'Saudi', 'Arabia']
['Nitric', 'oxide', ':', 'a', 'pro-inflammatory', 'mediator', 'in', 'lung', 'disease', '?']
['Surfactant', 'protein-D', 'and', 'pulmonary', 'host', 'defense']
['Role', 'of', 'endothelin-1', 'in', 'lung', 'disease']
['Gene', 'expression', 'in', 'epithelial', 'cells', 'in', 'response', 'to', 'pneumovirus', 'infection']
['Sequence', 'requirements', 'for', 'RNA', 'strand', 'transfer', 'during', 'nidovirus', 'discontinuous', 'subgenomic', 'RNA',
'synthesis']
['Debate', ':', 'Transfusing', 'to', 'normal', 'haemoglobin', 'levels', 'will', 'not', 'improve', 'outcome']
['The', '21st', 'International', 'Symposium', 'on', 'Intensive', 'Care', 'and', 'Emergency', 'Medicine', ',', 'Brussels',
',', 'Belgium', ',', '20-23', 'March', '2001']
['Heme', 'oxygenase-1', 'and', 'carbon', 'monoxide', 'in', 'pulmonary', 'medicine']
['Technical', 'Description', 'of', 'RODS', ':', 'A', 'Real-time', 'Public', 'Health', 'Surveillance', 'System']
['Conservation', 'of', 'polyamine', 'regulation', 'by', 'translational', 'frameshifting', 'from', 'yeast', 'to', 'mammals']
['Heterogeneous', 'nuclear', 'ribonucleoprotein', 'A1', 'regulates', 'RNA', 'synthesis', 'of', 'a', 'cytoplasmic', 'virus']
['A', 'Method', 'to', 'Identify', 'p62', "'s", 'UBA', 'Domain', 'Interacting', 'Proteins']
['Vaccinia', 'virus', 'infection', 'disrupts', 'microtubule', 'organization', 'and', 'centrosome', 'function']
['Multi-faceted', ',', 'multi-versatile', 'microarray', ':', 'simultaneous', 'detection', 'of', 'many', 'viruses', 'and', 'th
eir', 'expression', 'profiles']
```

Tokenization:

- Tokenization is used for obtaining the character sequence as "tokens". There are different types of tokenization supported by NLTK Library, like Whitespace Tokenization, Regular Expression Tokenization, Punctuation Tokenization.
- I have used "**RegexpTokenizer**" to extract regular expression and "**WordPunctTokenizer**" for removing punctuation marks from the words.
- To use this tokenizer, one should install **"punkt"** package from ntlk.

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
True
```

Using RegexpTokenizer :

- I have applied RegexpTokenizer on **'title'** and **'abstract'** columns. It extracted all regular expression from these two columns.

```
from nltk.tokenize import RegexpTokenizer

tokenizer = RegexpTokenizer('\w+|\$[\d\.]+|\S+\s*\n\s*\n\s*\w+|[^\w\s]+')
```

```
# Using Regular Expression form 'title' Column
for reg in d['title']:
    all_reg = tokenizer.tokenize(reg)
    print(all_reg)
```

```
['Clinical', 'features', 'of', 'culture', '-', 'proven', 'Mycoplasma', 'pneumoniae', 'infections', 'at', 'King', 'Abdulaziz',
'University', 'Hospital', ',', 'Jeddah', ',', 'Saudi', 'Arabia']
['Nitric', 'oxide', ':', 'a', 'pro', '-', 'inflammatory', 'mediator', 'in', 'lung', 'disease', '?']
['Surfactant', 'protein', '-', 'D', 'and', 'pulmonary', 'host', 'defense']
['Role', 'of', 'endothelin', '-', '1', 'in', 'lung', 'disease']
['Gene', 'expression', 'in', 'epithelial', 'cells', 'in', 'response', 'to', 'pneumovirus', 'infection']
['Sequence', 'requirements', 'for', 'RNA', 'strand', 'transfer', 'during', 'nidovirus', 'discontinuous', 'subgenomic', 'RNA',
'synthesis']
['Debate', ':', 'Transfusing', 'to', 'normal', 'haemoglobin', 'levels', 'will', 'not', 'improve', 'outcome']
['The', '21st', 'International', 'Symposium', 'on', 'Intensive', 'Care', 'and', 'Emergency', 'Medicine', ',', 'Brussels',
',', 'Belgium', ',', '20', '-', '23', 'March', '2001']
['Heme', 'oxygenase', '-', '1', 'and', 'carbon', 'monoxide', 'in', 'pulmonary', 'medicine']
['Technical', 'Description', 'of', 'RODS', ':', 'A', 'Real', '-', 'time', 'Public', 'Health', 'Surveillance', 'System']
['Conservation', 'of', 'polyamine', 'regulation', 'by', 'translational', 'frameshifting', 'from', 'yeast', 'to', 'mammals']
['Heterogeneous', 'nuclear', 'ribonucleoprotein', 'A1', 'regulates', 'RNA', 'synthesis', 'of', 'a', 'cytoplasmic', 'virus']
['A', 'Method', 'to', 'Identify', 'p62', "'", 's', 'UBA', 'Domain', 'Interacting', 'Proteins']
['Vaccinia', 'virus', 'infection', 'disrupts', 'microtubule', 'organization', 'and', 'centrosome', 'function']
['Multi', '-', 'faceted', ',', 'multi', '-', 'versatile', 'microarray', ':', 'simultaneous', 'detection', 'of', 'many', 'viru
ses', 'and', 'their', 'expression', 'profiles']
```

Using WordPunctTokenizer :

- I have applied WordPunctTokenizer on **'title'** and **'abstract'** columns. It has removed all punctuation marks from the words of these two columns.

```
from nltk.tokenize import WordPunctTokenizer

tokenizer = WordPunctTokenizer()

for pun in d['title']:
    all_punctuation = tokenizer.tokenize(pun)
    print(all_punctuation)
```

```
['Clinical', 'features', 'of', 'culture', '-', 'proven', 'Mycoplasma', 'pneumoniae', 'infections', 'at', 'King', 'Abdulaziz',
'University', 'Hospital', ',', 'Jeddah', ',', 'Saudi', 'Arabia']
['Nitric', 'oxide', ':', 'a', 'pro', '-', 'inflammatory', 'mediator', 'in', 'lung', 'disease', '?']
['Surfactant', 'protein', '-', 'D', 'and', 'pulmonary', 'host', 'defense']
['Role', 'of', 'endothelin', '-', '1', 'in', 'lung', 'disease']
['Gene', 'expression', 'in', 'epithelial', 'cells', 'in', 'response', 'to', 'pneumovirus', 'infection']
['Sequence', 'requirements', 'for', 'RNA', 'strand', 'transfer', 'during', 'nidovirus', 'discontinuous', 'subgenomic', 'RNA',
'synthesis']
['Debate', ':', 'Transfusing', 'to', 'normal', 'haemoglobin', 'levels', 'will', 'not', 'improve', 'outcome']
['The', '21st', 'International', 'Symposium', 'on', 'Intensive', 'Care', 'and', 'Emergency', 'Medicine', ',', 'Brussels',
',', 'Belgium', ',', '20', '-', '23', 'March', '2001']
['Heme', 'oxygenase', '-', '1', 'and', 'carbon', 'monoxide', 'in', 'pulmonary', 'medicine']
['Technical', 'Description', 'of', 'RODS', ':', 'A', 'Real', '-', 'time', 'Public', 'Health', 'Surveillance', 'System']
['Conservation', 'of', 'polyamine', 'regulation', 'by', 'translational', 'frameshifting', 'from', 'yeast', 'to', 'mammals']
['Heterogeneous', 'nuclear', 'ribonucleoprotein', 'A1', 'regulates', 'RNA', 'synthesis', 'of', 'a', 'cytoplasmic', 'virus']
['A', 'Method', 'to', 'Identify', 'p62', "'", 's', 'UBA', 'Domain', 'Interacting', 'Proteins']
['Vaccinia', 'virus', 'infection', 'disrupts', 'microtubule', 'organization', 'and', 'centrosome', 'function']
['Multi', '-', 'faceted', ',', 'multi', '-', 'versatile', 'microarray', ':', 'simultaneous', 'detection', 'of', 'many', 'viru
ses', 'and', 'their', 'expression', 'profiles']
```

Normalization:

- Normalization is the process which transforming the text into standard form, for that I have implemented the conversion of words from upper case into lower case, and removing stop words.

Convert into Lower Case :

- **lower()** is used to convert the capital letters into small one.
- I have applied this conversion to **"authors"** and **"journal"** columns. As they have capital letters.

```
for sentences in d['journal']:
    lower_journal = sentences.lower()
    print(lower_journal)
```

```
bmc infect dis
respir res
respir res
respir res
respir res
the embo journal
crit care
crit care
respir res
journal of the american medical informatics association
embo j
the embo journal
biol proced online
the embo journal
retrovirology
crit care
bmc public health
respir res
bmc genomics
```

Removing the Stop Words:

- In the language the Stop words have no meaning, so it is better to extract from the text.
- To remove the stop words from the text before one should import **"stopwords"** package.

```
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
True
```

- I have removed stop words from **'title', 'abstract'** and **'authors'** columns.

```
# Remove StopWords form 'abstract' Column
for stops in d['abstract']:
    word_tokens = word_tokenize(stops)

removing_stopwords = [word for word in word_tokens if word not in stop_words]
print (removing_stopwords)
```

```
['Motivation', ':', 'Laboratory', 'RNA', 'structure', 'determination', 'demanding', 'costly', 'thus', ',', 'computational', 'st
ructure', 'prediction', 'important', 'task', '.', 'Single', 'sequence', 'methods', 'RNA', 'secondary', 'structure', 'predictio
n', 'limited', 'accuracy', 'underlying', 'folding', 'model', ',', 'structure', 'supported', 'family', 'evolutionarily', 'relate
d', 'sequences', ',', 'one', 'confident', 'prediction', 'accurate', '.', 'RNA', 'pseudoknots', 'functional', 'elements', ',',
'highly', 'conserved', 'structures', '.', 'However', ',', 'comparative', 'structure', 'prediction', 'methods', 'handle', 'pseud
oknots', 'due', 'computational', 'complexity', '.', 'Results', ':', 'A', 'comparative', 'pseudoknot', 'prediction', 'method',
'called', 'DotKnot-PW', 'introduced', 'based', 'structural', 'comparison', 'secondary', 'structure', 'elements', 'H-type', 'pse
udoknot', 'candidates', '.', 'DotKnot-PW', 'outperforms', 'methods', 'literature', 'hand-curated', 'test', 'set', 'RNA', 'struc
tures', 'experimental', 'support', '.', 'Availability', ':', 'DotKnot-PW', 'RNA', 'structure', 'test', 'set', 'available', 'we
b', 'site', 'http', ':', '//dotknot.csse.uwa.edu.au/pw', '.', 'Contact', ':', 'janaspe', '@', 'csse.uwa.edu.au', 'Supplementar
y', 'information', ':', 'Supplementary', 'data', 'available', 'Bioinformatics', 'online', '.']
```

4) Selecting Keywords:

- Keyword filter is used to determine the keywords from the document with maximum weightage, and it is calculated using TF * IDF measure.

- TF (Term Frequency) is the count in document. As frequency will increase then word's weight is also increase. *IDF (Inverse Document Frequency) is used to measure the word which appears in dataset. And it is multiplied with TF value.*

- *As I consider four columns 'title', 'abstract', 'authors' and 'journal' from the dataset. The words from these columns were stored in the list and individual weight will be calculated. To count the vocabulary from these columns I have used* ***"CountVectorizer"***

- *Import CountVectorizer from the sklearn library to calculate the vocabulary.*

```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_df=0.85,stop_words=words)
word_count_vector=cv.fit_transform(docs)
```

```
C:\Users\user\miniconda3\envs\ce802\lib\site-packages\sklearn\feature_extraction\text.py:301: UserWarning: Your stop_words may
be inconsistent with your preprocessing. Tokenizing the stop words generated tokens ['amitava', 'availability', 'bioinformatic
s', 'contact', 'datta', 'dotknotpw', 'however', 'htype', 'jana', 'jbioinformatics', 'laboratory', 'michael', 'onlinesperschneid
er', 'predicting', 'results', 'rna', 'sequencesmotivation', 'single', 'supplementary', 'wise'] not in stop_words.
  'stop_words.' % sorted(inconsistent))
```

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
tfidf_matrix = tfidf_transformer.fit(word_count_vector)
```

```
feature_names = cv.get_feature_names()
vector = docs[0]
tf_idf_vector=tfidf_transformer.transform(cv.transform([vector]))
```

- *The following is the code to calculate the weightage of all the individual keywords. I have considered 1000 words from the documents.*

```
#sort the tf-idf vectors by descending order of scores
sorted_items=sort(tf_idf_vector.tocoo())
#extract only the top n; n here is 1000
keywords=extract(feature_names,sorted_items,1000)

print(vector)

for k in keywords :
    print(k,keywords[k])
```

```
ients with comorbidities was high.madani, tariq a; al-ghamdi, aisha abmc infect dis
patients 0.31
pneumoniae 0.302
comorbidities 0.288
pneumonia 0.252
common 0.185
were 0.181
had 0.162
died 0.161
most 0.159
infections 0.155
jeddah 0.153
crepitations 0.153
abdulaziz 0.153
saudi 0.144
mycoplasma 0.133
king 0.126
proven 0.121
with 0.118
```

5) Stemming or Morphological Analysis:

Stemming:

- Stemming is the process to reduce the suffix and prefix from the word. For Example, If Males will be there then it will be reduced in 'male'.
- I used **porterstemmer** library from NLTK package, which has all stemming rules. And I also used **wordtokenize** to work with sentence.
- I have applied stemming on words and sentences with **'title'** and **'abstract'** columns.

Using Words:

```
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
porter_stemmer = PorterStemmer()
```

```
words = d['title']
for w in words:
    print(w, " : ",porter_stemmer.stem(w))
```

```
Clinical features of culture-proven Mycoplasma pneumoniae infections at King Abdulaziz University Hospital, Jeddah, Saudi Ara
bia  :  clinical features of culture-proven mycoplasma pneumoniae infections at king abdulaziz university hospital, jeddah, s
audi arabia
Nitric oxide: a pro-inflammatory mediator in lung disease?  :  nitric oxide: a pro-inflammatory mediator in lung disease?
Surfactant protein-D and pulmonary host defense  :  surfactant protein-d and pulmonary host defens
Role of endothelin-1 in lung disease  :  role of endothelin-1 in lung diseas
Gene expression in epithelial cells in response to pneumovirus infection  :  gene expression in epithelial cells in response
to pneumovirus infect
Sequence requirements for RNA strand transfer during nidovirus discontinuous subgenomic RNA synthesis  :  sequence requiremen
ts for rna strand transfer during nidovirus discontinuous subgenomic rna synthesi
Debate: Transfusing to normal haemoglobin levels will not improve outcome  :  debate: transfusing to normal haemoglobin level
s will not improve outcom
The 21st International Symposium on Intensive Care and Emergency Medicine, Brussels, Belgium, 20-23 March 2001  :  the 21st i
nternational symposium on intensive care and emergency medicine, brussels, belgium, 20-23 march 2001
Heme oxygenase-1 and carbon monoxide in pulmonary medicine  :  heme oxygenase-1 and carbon monoxide in pulmonary medicin
Technical Description of RODS: A Real-time Public Health Surveillance System  :  technical description of rods: a real-time p
ublic health surveillance system
Conservation of polyamine regulation by translational frameshifting from yeast to mammals  :  conservation of polyamine regul
ation by translational frameshifting from yeast to mamm
```

Using Sentences:

- To stem the words and sentences one should to download the **"wordnet"** package from the NLTK library.

```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
True
```

```
from nltk.stem.wordnet import WordNetLemmatizer
wnl = WordNetLemmatizer()
```

- Considered "title" column for sentence stemming.

```
for sentence in d['title']:
    words = word_tokenize(sentence)

    for w in words:
          print(w, " : ", porter_stemmer.stem(w),wnl.lemmatize(w))
```

```
Clinical  :  clinic Clinical
features  :  featur feature
of  :  of of
culture-proven  :  culture-proven culture-proven
Mycoplasma  :  mycoplasma Mycoplasma
pneumoniae  :  pneumonia pneumoniae
infections  :  infect infection
at  :  at at
King  :  king King
Abdulaziz  :  abdulaziz Abdulaziz
University  :  univers University
Hospital  :  hospit Hospital
,  :  , ,
Jeddah  :  jeddah Jeddah
,  :  , ,
Saudi  :  saudi Saudi
Arabia  :  arabia Arabia
Nitric  :  nitric Nitric
```

Lemmatization:

- Lemmatization is best process to reduce the words from its root. It is very efficient compared to stemming because it resolves the word to their dictionary meaning.

- I have used **wordnetlemmatizer** package from NLTK library. I used to lemmatize function which gives the root from the words.

Why Use Lemmatization instead of Stemming:

- Stemming gives kind of word by removing surface or preface that is not valid word in the dictionary, while Lemmatization gives the most appropriate word.

- **Example:** This: Thi : This

Using Words:

- I have considered **"title"** and **"abstract"** columns for lemmatize the words.

```
words = d['title']
for w in words:
    print(w, " : ",wnl.lemmatize(w))
```

```
Clinical features of culture-proven Mycoplasma pneumoniae infections at King Abdulaziz University Hospital, Jeddah, Saudi Ara
bia  :  Clinical features of culture-proven Mycoplasma pneumoniae infections at King Abdulaziz University Hospital, Jeddah, S
audi Arabia
Nitric oxide: a pro-inflammatory mediator in lung disease?  :  Nitric oxide: a pro-inflammatory mediator in lung disease?
Surfactant protein-D and pulmonary host defense  :  Surfactant protein-D and pulmonary host defense
Role of endothelin-1 in lung disease  :  Role of endothelin-1 in lung disease
Gene expression in epithelial cells in response to pneumovirus infection  :  Gene expression in epithelial cells in response
to pneumovirus infection
Sequence requirements for RNA strand transfer during nidovirus discontinuous subgenomic RNA synthesis  :  Sequence requiremen
ts for RNA strand transfer during nidovirus discontinuous subgenomic RNA synthesis
Debate: Transfusing to normal haemoglobin levels will not improve outcome  :  Debate: Transfusing to normal haemoglobin level
s will not improve outcome
The 21st International Symposium on Intensive Care and Emergency Medicine, Brussels, Belgium, 20-23 March 2001  :  The 21st I
nternational Symposium on Intensive Care and Emergency Medicine, Brussels, Belgium, 20-23 March 2001
Heme oxygenase-1 and carbon monoxide in pulmonary medicine  :  Heme oxygenase-1 and carbon monoxide in pulmonary medicine
Technical Description of RODS: A Real-time Public Health Surveillance System  :  Technical Description of RODS: A Real-time P
ublic Health Surveillance System
Conservation of polyamine regulation by translational frameshifting from yeast to mammals  :  Conservation of polyamine regul
ation by translational frameshifting from yeast to mammals
```

Using Sentences:

- I have considered **"title"** and **"abstract"** columns for lemmatize the sentences.

```
for sentence in d['title']:
    input_str = word_tokenize(sentence)

    for word in input_str:
        print(wnl.lemmatize(word))
```

```
Clinical
feature
of
culture-proven
Mycoplasma
pneumoniae
infection
at
King
Abdulaziz
University
Hospital
,
Jeddah
,
Saudi
Arabia
Nitric
oxide
```

6) Searching :

- *After creating an index it's time to searching those documents which I have indexed! Once Index has been created, then We can apply different criteria for searching the documents. We can search by ID or we can pass request body. We can use DSL Query for searching purpose.*
- *The following is the code search by index without query.*

```
# ====== Searching Documents ====== #

# Retrieving all documents in index (no query given)
documents = es.search(index='covid',body={})['hits']['hits']
df = pd.DataFrame(documents)
print(df)
```

```
  _index       _type                    _id  _score  \
0  covid  covid_data  RqRj-3cBpKvraAqVbPL2     1.0
1  covid  covid_data  R6Rj-3cBpKvraAqVbPL2     1.0
2  covid  covid_data  SqRj-3cBpKvraAqVbPL2     1.0
3  covid  covid_data  UaRj-3cBpKvraAqVbPL2     1.0
4  covid  covid_data  YKRj-3cBpKvraAqVbPL2     1.0
5  covid  covid_data  Z6Rj-3cBpKvraAqVbPL2     1.0
6  covid  covid_data  aaRj-3cBpKvraAqVbPL2     1.0
7  covid  covid_data  aqRj-3cBpKvraAqVbPL2     1.0
8  covid  covid_data  bqRj-3cBpKvraAqVbPL2     1.0
9  covid  covid_data  caRj-3cBpKvraAqVbPL2     1.0

                                              _source
0  {'title': 'debate: transfusing to normal haemo...
1  {'title': 'the 21st international symposium on...
2  {'title': 'conservation of polyamine regulatio...
3  {'title': 'protection of pulmonary epithelial ...
4  {'title': 'globalization and health', 'abstrac...
5  {'title': 'evaluation of potential reference g...
6  {'title': 'bioethical implications of globaliz...
7  {'title': 'public awareness of risk factors fo...
8  {'title': 'local public health workers' percep...
9  {'title': 'markers of exacerbation severity in...
```

- *The following is the code search by index and I passed query which match all the documents.*

```
# check data is in there, and structure in there
documents2 = es.search(body={"query": {"match_all": {}}}, index = 'covid')
df2 = pd.DataFrame(documents2)
print(df2)
```

```
            took  timed_out  _shards  \
total         32      False      5.0
successful    32      False      5.0
skipped       32      False      0.0
failed        32      False      0.0
max_score     32      False      NaN
hits          32      False      NaN

                                                     hits
total                                                 954
successful                                            NaN
skipped                                               NaN
failed                                                NaN
max_score                                               1
hits        [{'_index': 'covid', '_type': 'covid_data', '_...
```

- *The following is the code search by index and I passed query which match "title = clinical" it gives matches the documents.*

```
# Retrieving documents in index that match a title
documents2 = es.search(index='covid',body={"query":{"term":{"title" : "clinical"}}})['hits']['hits']
df2 = pd.DataFrame(documents2)
print(df2)
```

```
  _index       _type                   _id    _score  \
0  covid  covid_data  bKXy-3cBpKvraAqV-gSh  4.501509
1  covid  covid_data  aaXy-3cBpKvraAqV-gIG  4.286892
2  covid  covid_data  rqXy-3cBpKvraAqV-gOh  4.204507
3  covid  covid_data  n6Xy-3cBpKvraAqV-gEG  4.160217
4  covid  covid_data  raXy-3cBpKvraAqV-gOh  3.942394
5  covid  covid_data  5qXy-3cBpKvraAqV-gIG  3.823223
6  covid  covid_data  rqXy-3cBpKvraAqV-gSh  3.654502
7  covid  covid_data  oKXy-3cBpKvraAqV-gEG  3.601176
8  covid  covid_data  m6Xy-3cBpKvraAqV-gSh  3.594622
9  covid  covid_data  h6Xy-3cBpKvraAqV-gOh  3.546718

                                      _source
0  {'title': 'clinical review: special population...
1  {'title': 'clinical review: primary influenza ...
2  {'title': 'clinical review: idiopathic pulmona...
3  {'title': 'clinical review: update of avian in...
4  {'title': 'clinical aspects and cytokine respo...
5  {'title': 'clinical factors associated with se...
6  {'title': 'influenza a: from highly pathogenic...
7  {'title': 'clinical review: mass casualty tria...
8  {'title': 'clinical characteristics and outcom...
9  {'title': 'epidemiological and clinical charac...
```

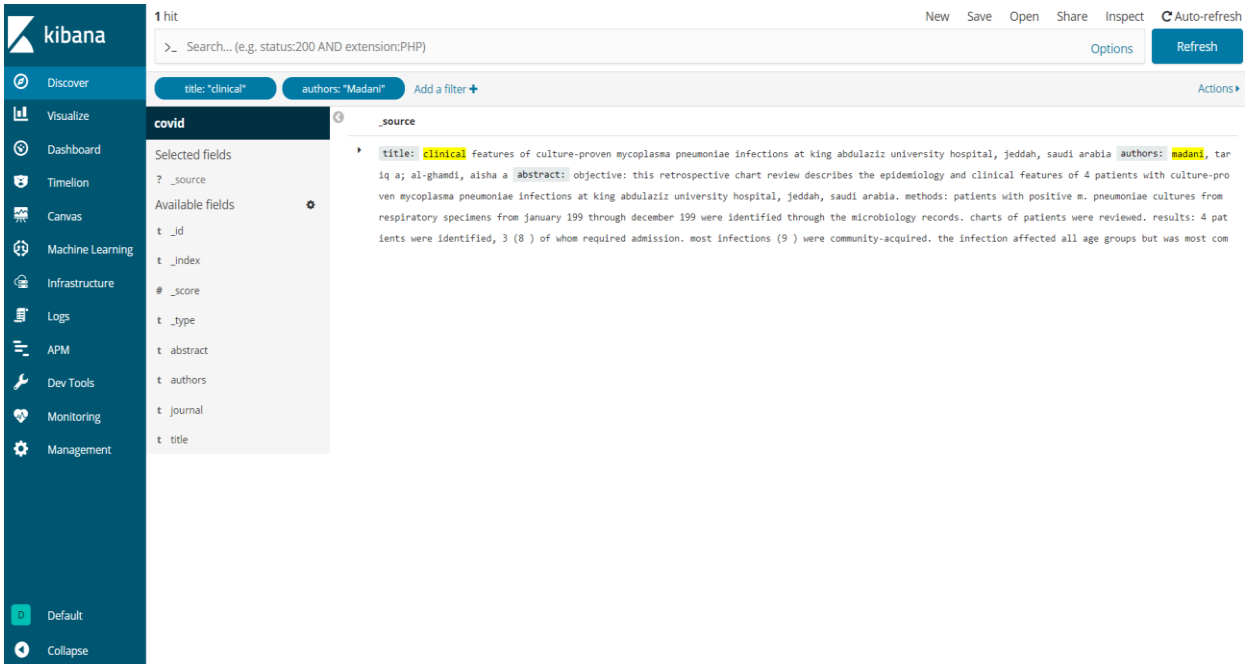
- *The following query retunes the data which has value abstract = "objective ".*

```
# Retrieving documents in index that match a abstract
documents2 = es.search(index='covid',body={"query":{"term":{"abstract" : "objective"}}})['hits']['hits']
df2 = pd.DataFrame(documents2)
print(df2)
```

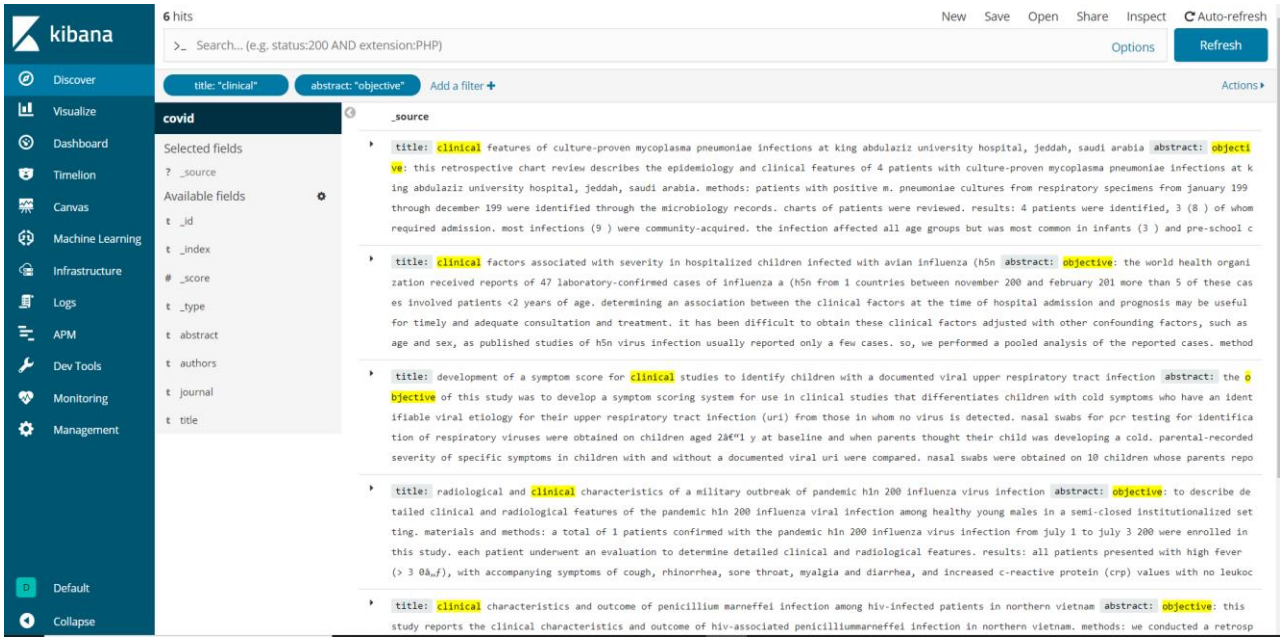
```
  _index       _type                   _id    _score  \
0  covid  covid_data  3V4EAngBsLn8hPS_Kbqy  4.360685
1  covid  covid_data  HV4EAngBsLn8hPS_Kbqy  4.252325
2  covid  covid_data  Ll4EAngBsLn8hPS_KLnC  4.207901
3  covid  covid_data  Fl4EAngBsLn8hPS_Kbuy  3.889466
4  covid  covid_data  9l4EAngBsLn8hPS_Kbqy  3.640916
5  covid  covid_data  rl4EAngBsLn8hPS_KLnD  3.559970
6  covid  covid_data  r14EAngBsLn8hPS_KLjC  3.535354
7  covid  covid_data  y14EAngBsLn8hPS_KLnD  3.518721
8  covid  covid_data  K14EAngBsLn8hPS_Kbqy  3.473717
9  covid  covid_data  fl4EAngBsLn8hPS_KLnD  3.301905

                                      _source
0  {'title': 'reliability and external validity o...
1  {'title': 'standardization of methods for earl...
2  {'title': 'outdoor environments and human path...
3  {'title': 'genome stability of pandemic influe...
4  {'title': 'a scientometric analysis of indian ...
5  {'title': 'environmental factors preceding ill...
6  {'title': 'multiorgan failure due to hemophago...
7  {'title': 'pathological and ultrastructural an...
8  {'title': 'seasonal distribution of active sys...
9  {'title': 'radiological and clinical character...
```

- *I have used Kibana to visualize the data, and the following few screenshots are there which I have taken from Kibana for searching purpose.*

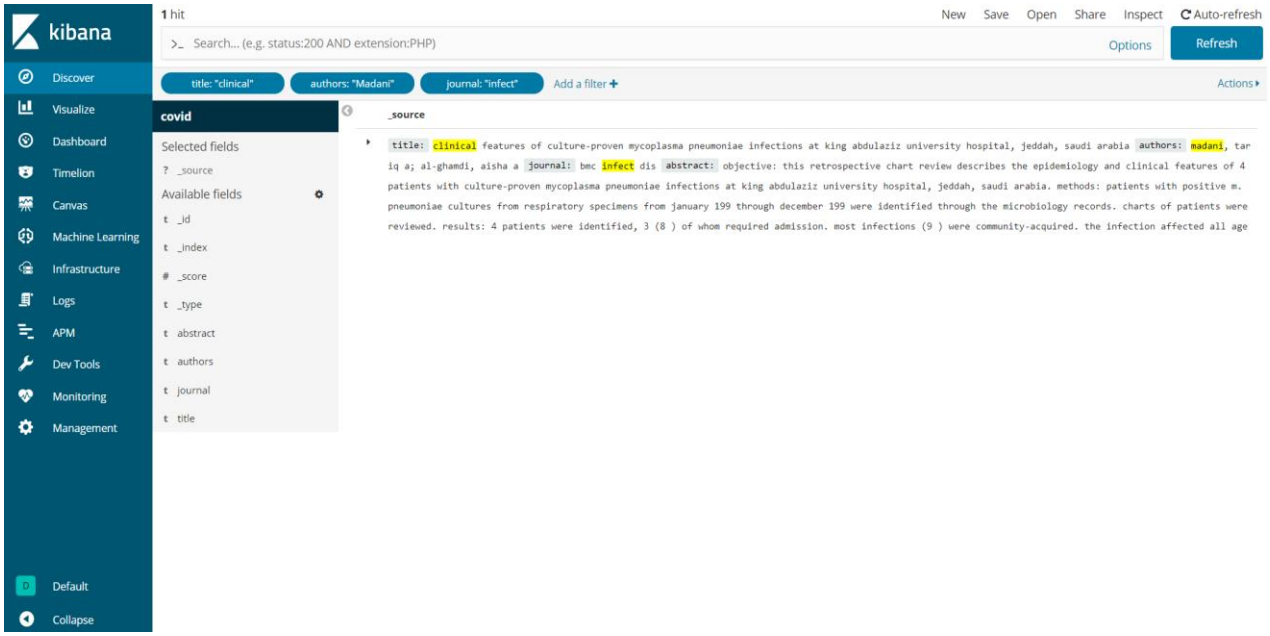
- When title =" clinical" it will retrieve the data which has matching title.



- When title =" clinical" and abstract = "objective" it will retrieve the data which has matching title and abstract.



- When title =" clinical", author = "Madani" and journal="infect" it will retrieve the data which has matching title, author and journal.


kibana
Discover
Visualize
Dashboard
Timelion
Canvas
Machine Learning
Infrastructure
Logs
APM
Dev Tools
Monitoring
Management
Default
Collapse
1 hit
New
Save
Open
Share
Inspect
Auto-refresh
Search... (e.g. status:200 AND extension:PHP)
Options
Refresh
title: "clinical"
authors: "Madani"
journal: "infect"
Add a filter
Actions
covid
Selected fields
? _source
Available fields
t _id
t _index
_score
t _type
t abstract
t authors
t journal
t title
_source
title: clinical features of culture-proven mycoplasma pneumoniae infections at king abdulaziz university hospital, jeddah, saudi arabia authors: madani, tariq a; al-ghamdi, aisha a journal: bmc infect dis abstract: objective: this retrospective chart review describes the epidemiology and clinical features of 4 patients with culture-proven mycoplasma pneumoniae infections at king abdulaziz university hospital, jeddah, saudi arabia. methods: patients with positive m. pneumoniae cultures from respiratory specimens from january 199 through december 199 were identified through the microbiology records. charts of patients were reviewed. results: 4 patients were identified, 3 (8) of whom required admission. most infections (9) were community-acquired. the infection affected all age