CE706 - Information Retrieval 2021

Assignment 1

Student Id: 2004458

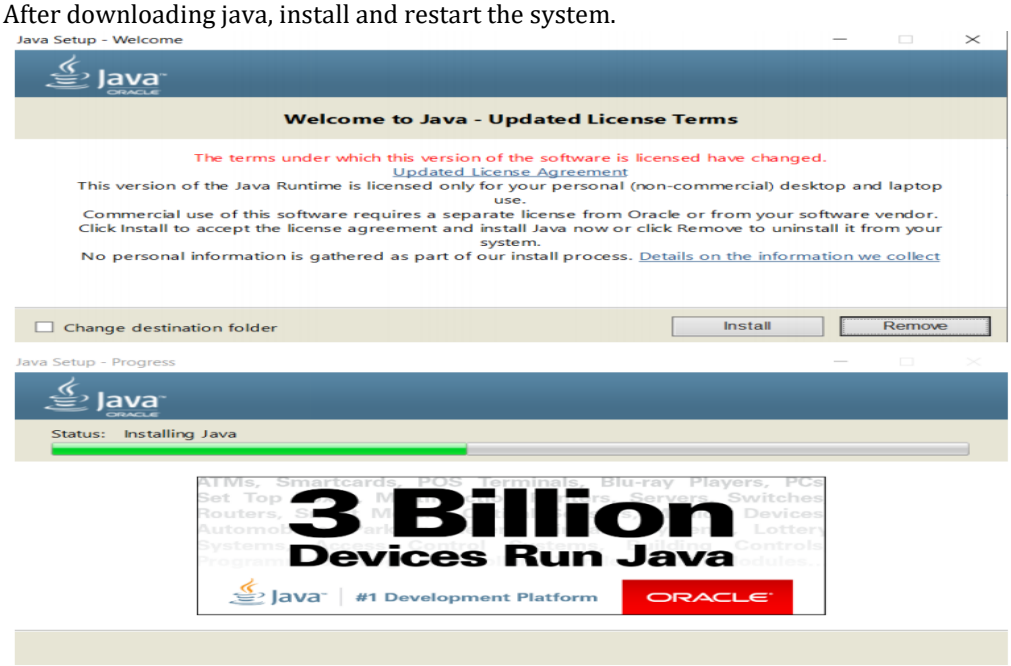
1. Instructions for running the system:

Prerequisites:

- Windows10 operating system, Python 3.8, Python libraries namely, NLTK, sklearn, Pandas, NumPy, re.

```
import pandas as pd
import numpy as np
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfTransformer
```

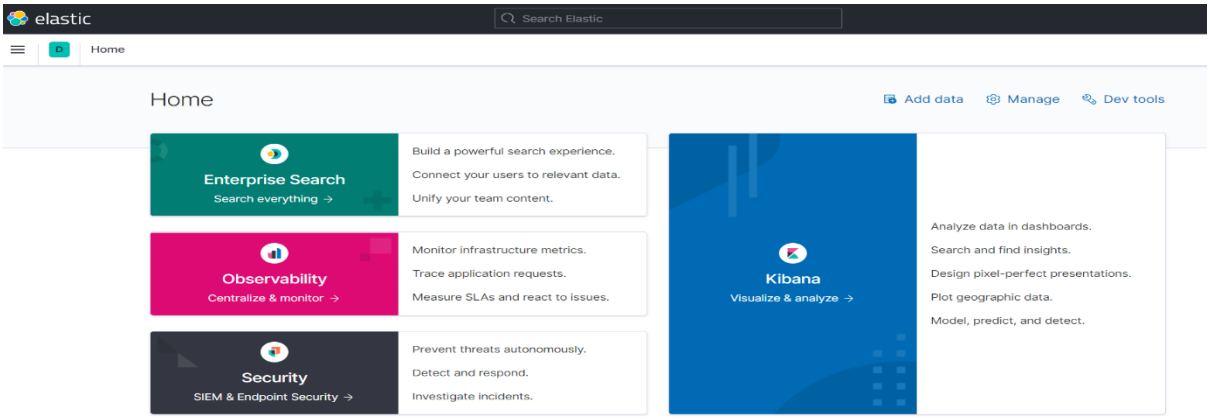
- Download these software: Elasticsearch 7.11.1, Kibana7.11.1, Logstash 7.11.1, java version 1.8.0_281.
- After downloading java, install and restart the system.



- After downloading Elasticsearch 7.11.1, unzip it and go to bin folder. In the bin folder, select elastic search windows batch file, right click and run as administrator. Now, the Elasticsearch runs on port 9002. Open any browser and paste http://localhost:9200/ to check if Elasticsearch is running. If the Elasticsearch is installed correctly, the following message will be displayed:

```
{
  "name" : "LAPTOP-BHI84IJF",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "IPVVKX9rRNKkbHGEYEVg2Q",
  "version" : {
    "number" : "7.11.1",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "ff17057114c2199c9c1bbecc727003a907c0db7a",
    "build_date" : "2021-02-15T13:44:09.394032Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```


- After downloading Kibana 7.11.1, unzip it and go to bin folder. In the bin folder, select Kibana windows batch file, right click on it and run as administrator. Paste http://localhost:5601/ in the browser, if the Kibana is installed successfully it looks like this:



- Don't close Elasticsearch and Kibana, let them run in background. Just minimize the command prompt window.
- After downloading the Logstash 7.11.1, extract the folder. We can run Logstash from command line, which is discussed in **Indexing** section.

Dataset Acquisition and Data Description:

- The dataset used for this assignment is called 'CORD-19' (COVID-19 Open Research Dataset) dataset. CORD-19 is a collection of 400,000 scholarly articles about COVID-19, SARS-CoV-2, and related corona viruses. This is a freely available dataset on Kaggle. The 'metadata.csv' file from Kaggle is used for this assignment. The link to download the dataset is given below:
- https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge?select=metadata.csv
- The dataset has 467521 rows and 19 columns. Each row has information such as title, abstract, author names etc. about each scholarly article.
- After downloading the data, it is read into pandas' data frame. The dataset looks like this:

```
covid_df = pd.read_csv("metadata\metadata.csv",low_memory=False)
covid_df.head(2)
```

	cord_uid	sha	source_x	title	doi	pmcid	pubmed_id	license	abstract	publish_time	a
0	ug7v899j	d1aafb70c066a2068b02786f8929fd9c900897fb	PMC	Clinical features of culture-proven Mycoplasma...	10.1186/1471-2334-1-6	PMC35282	11472636	no-cc	OBJECTIVE: This retrospective chart review des...	2001-07-04	M T G A
1	02tnwd4m	6b0567729c2143a66d737eb0a2f63f2dce2e5a7d	PMC	Nitric oxide: a pro-inflammatory mediator in l...	10.1186/rr14	PMC59543	11667967	no-cc	Inflammatory diseases of the respiratory tract...	2000-08-15	v Ei

```
columnnames = covid_df.columns
columnnames
```

```
Index(['cord_uid', 'sha', 'source_x', 'title', 'doi', 'pmcid', 'pubmed_id',
       'license', 'abstract', 'publish_time', 'authors', 'journal', 'mag_id',
       'who_covidence_id', 'arxiv_id', 'pdf_json_files', 'pmc_json_files',
       'url', 's2_id'],
      dtype='object')
```

```
covid_df.shape
```

```
(467521, 19)
```

- Then dataset is checked for any missing values. First 1300 rows of the data frame are selected and checked for missing values; some columns have missing values which can be seen below:

```
## Selecting 1300 docs to start with
docs = covid_df.head(1300)
len(docs)
```

```
1300
```

```
docs.isnull().sum()
```

```
cord_uid              0
sha                  51
source_x              0
title                 0
doi                   0
pmcid                 0
pubmed_id             0
license               0
abstract             59
publish_time          0
authors              20
journal               0
mag_id             1300
who_covidence_id   1300
arxiv_id           1300
pdf_json_files       51
pmc_json_files       46
url                   0
s2_id              1300
dtype: int64
```

- ‘mag_id’, ‘who_covidence_id’, ‘arxiv_id’, ‘s2_id’ columns have all missing values. So, these columns are dropped from data frame.
- Then from columns ‘sha’, abstract’,’pdf_json_files’,’pmc_json_files’, rows with no missing values are selected.

```
docs = docs.dropna(axis=1, how='all') # dropping the columns with all missing values
docs.columns
```

```
Index(['cord_uid', 'sha', 'source_x', 'title', 'doi', 'pmcid', 'pubmed_id',
       'license', 'abstract', 'publish_time', 'authors', 'journal',
       'pdf_json_files', 'pmc_json_files', 'url'],
      dtype='object')
```

```
# mag_id,who_covidence_id,arxiv_id,s2_id theses columns have all missing values so those 4 columns are dropped
```

```
docs_with_abstract = np.where(docs.abstract.notnull()) # selecting rows with no abstract missing
docs = docs.iloc[docs_with_abstract]
```

```
docs_with_sha = np.where(docs.sha.notnull())
docs = docs.iloc[docs_with_sha]
```

```
docs_with_authors = np.where(docs.authors.notnull())
docs = docs.iloc[docs_with_authors]
```

```
docs_with_pdf_json_files = np.where(docs.pdf_json_files.notnull())
docs_with_pmf_json_files = np.where(docs.pmc_json_files.notnull())
docs = docs.iloc[docs_with_pdf_json_files]
docs = docs.iloc[docs_with_pmf_json_files]
```

```
len(docs)
```

```
1186
```

- Now, the data frame 'docs' doesn't have any missing values and has a total of 15 columns. From this data frame, first 1000 rows are selected and saved to a data frame called 'docs'.

```
docs = docs.head(1000) # These 1000 documents are indexed using elasticsearch
docs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 1092
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   cord_uid        1000 non-null   object
 1   sha             1000 non-null   object
 2   source_x        1000 non-null   object
 3   title           1000 non-null   object
 4   doi             1000 non-null   object
 5   pmcid           1000 non-null   object
 6   pubmed_id       1000 non-null   object
 7   license         1000 non-null   object
 8   abstract        1000 non-null   object
 9   publish_time    1000 non-null   object
 10  authors         1000 non-null   object
 11  journal         1000 non-null   object
 12  pdf_json_files  1000 non-null   object
 13  pmc_json_files  1000 non-null   object
 14  url             1000 non-null   object
dtypes: object(15)
```

- This data frame is saved to a .csv file and this file is used for indexing.

```
docs.to_csv("CORD-19_1000docs.csv",index=False)
```

2. Indexing:

- Indexing is done using Logstash 7.11.1. Logstash is a server-side data processing pipeline that ingests data from multiple sources, transforms it and sends it to a "stash" like Elasticsearch.
- To configure Logstash, first 'logstash.conf' file created. This is how the 'logstash.conf' file looks like:

```
input {
  file {
      path => "C:/Users/jhans/OneDrive/Documents/CORD-19_
1000docs.csv"
      start_position => "beginning"
      sincedb_path => "NULL"
      }
}
filter {
  csv{
      separator => ","
      columns => ["cord_uid", "sha", "source_x", "title", "doi",
"pmcid", "pubmed_id","license", "abstract", "publish_time",
"authors", "journal", "pdf_json_files", "pmc_json_files","url"]
     }
}
output {
  elasticsearch
      {
      hosts => ["http://localhost:9200"]
      index => "covid-19"
      }
      stdout {}
}
```

The Logstash event processing pipeline has three stages: input-> filters -> outputs.

Inputs: An input plugin is used to get the data into Logstash. The input plugin has a plugin called **file,** that enables streaming of events from file. The following configuration are set using file plugin:

- **path:** The path to the file to use as an input.
- **start_position:** Specifying Logstash to read from begging of the file**.**
- **sincedb_path:** Path of sincedb database file, which keeps track of the current position of monitored log files that will be written to disk. Since I used Windows 10 operating system I set this value to 'NULL'.

Filters: Filters are intermediary processing devices in the Logstash pipeline.

The **CSV** filter takes an event containing CSV data, parses it, and stores it as individual fields with specified filed names.

- **columns:** A list of column names as appears in the given file.
- **separator:** Define the column separator, since we are using csv file, the separator is comma (',').

Outputs: Outputs are the final stage of the Logstash pipeline.

Elasticsearch output plugin used to send data into Elasticsearch, this enables using of Kibana interface to analyze data transformed by Logstash.

- **hosts:** The URL to Elasticsearch port.
- **Index**: The index to write events to.

stdout displays the process in command line.

The following command is used to run Logstash from command line: **bin/logstash -f logstash.conf**

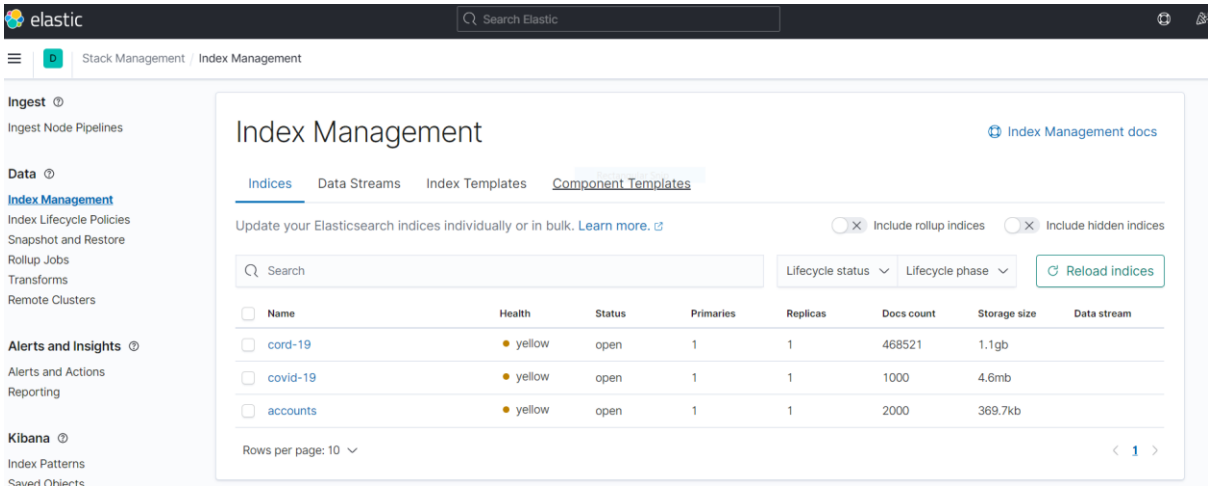
- Open command line, go to directory where Logstash bin file is located.
- Then type the above command and press enter. This starts indexing of documents.

```
C:\Users\jhans\OneDrive\Documents>C:\Users\jhans\OneDrive\Documents\logstash-7.11.1\bin\logstash -f C:\Users\jhans\OneDrive\Documents\logstash.conf
```

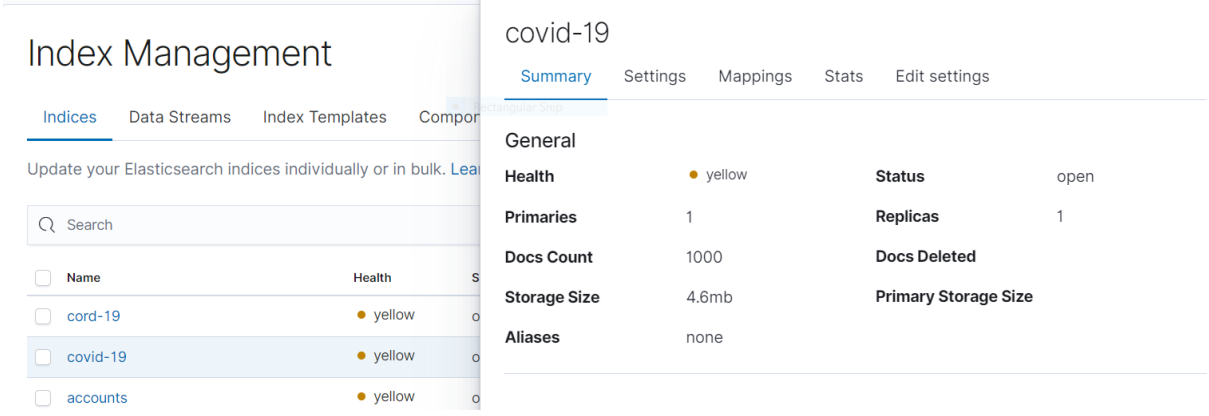
- The indexed documents look like this and the output is sent directly to Elastic search.

```
{
    "pdf_json_files" => "document_parses/pdf_json/df783d511b145a10e7f609a87392eb50799d2b2b.json",
           "license" => "cc-by",
           "message" => "0jx6mwiw,df783d511b145a10e7f609a87392eb50799d2b2b,PMC,\"NOA36 Protein Contains a Highly Conserved Nucleolar Localization Signal Capable of Directing Functional Proteins to the Nucleolus,
 in Mammalian Cells\",10.1371/journal.pone.0059065,PMC3596294,23516598,cc-by,\"NOA36/ZNF330 is an evolutionarily well-preserved protein present in the nucleolus and mitochondria of mammalian cells. We have previ
ously reported that the pro-apoptotic activity of this protein is mediated by a characteristic cysteine-rich domain. We now demonstrate that the nucleolar localization of NOA36 is due to a highly-conserved nucle
olar localization signal (NoLS) present in residues 1?33. This NoLS is a sequence containing three clusters of two or three basic amino acids. We fused the amino terminal of NOA36 to eGFP in order to characteriz
e this putative NoLS. We show that a cluster of three lysine residues at positions 3 to 5 within this sequence is critical for the nucleolar localization. We also demonstrate that the sequence as found in human
is capable of directing eGFP to the nucleolus in several mammal, fish and insect cells. Moreover, this NoLS is capable of specifically directing the cytosolic yeast enzyme polyphosphatase to the target of the nu
cleolus of HeLa cells, wherein its enzymatic activity was detected. This NoLS could therefore serve as a very useful tool as a nucleolar marker and for directing particular proteins to the nucleolus in distant a
nimal species.\",2013-03-13,\"de Melo, Ivan S.; Jimenez-Nuñez, Maria D.; Iglesias, Concepción; Campos-Caro, Antonio; Moreno-Sanchez, David; Ruiz, Felix A.; Bolívar, Jorge\",PLoS One,document_parses/pdf_json/df78
3d511b145a10e7f609a87392eb50799d2b2b.json,document_parses/pmc_json/PMC3596294.xml.json,https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3596294/\r",
          "source_x" => "PMC",
           "journal" => "PLoS One",
          "@version" => "1",
               "doi" => "10.1371/journal.pone.0059065",
      "publish_time" => "2013-03-13",
              "host" => "LAPTOP-8HI84IJF",
               "sha" => "df783d511b145a10e7f609a87392eb50799d2b2b",
             "pmcid" => "PMC3596294",
              "path" => "C:/Users/jhans/Information Rereieval/CORD-19_1000docs.csv",
         "pubmed_id" => "23516598",
          "abstract" => "NOA36/ZNF330 is an evolutionarily well-preserved protein present in the nucleolus and mitochondria of mammalian cells. We have previously reported that the pro-apoptotic activity of this
 protein is mediated by a characteristic cysteine-rich domain. We now demonstrate that the nucleolar localization of NOA36 is due to a highly-conserved nucleolar localization signal (NoLS) present in residues 1?
33. This NoLS is a sequence containing three clusters of two or three basic amino acids. We fused the amino terminal of NOA36 to eGFP in order to characterize this putative NoLS. We show that a cluster of three
lysine residues at positions 3 to 5 within this sequence is critical for the nucleolar localization. We also demonstrate that the sequence as found in human is capable of directing eGFP to the nucleolus in sever
al mammal, fish and insect cells. Moreover, this NoLS is capable of specifically directing the cytosolic yeast enzyme polyphosphatase to the target of the nucleolus of HeLa cells, wherein its enzymatic activity
was detected. This NoLS could therefore serve as a very useful tool as a nucleolar marker and for directing particular proteins to the nucleolus in distant animal species.",
           "authors" => "de Melo, Ivan S.; Jimenez-Nuñez, Maria D.; Iglesias, Concepción; Campos-Caro, Antonio; Moreno-Sanchez, David; Ruiz, Felix A.; Bolívar, Jorge",
    "pmc_json_files" => "document_parses/pmc_json/PMC3596294.xml.json",
             "title" => "NOA36 Protein Contains a Highly Conserved Nucleolar Localization Signal Capable of Directing Functional Proteins to the Nucleolus, in Mammalian Cells",
         "cord_uid" => "0jx6mwiw",
               "url" => "https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3596294/",
        "@timestamp" => 2021-03-02T00:37:43.421Z
}
```

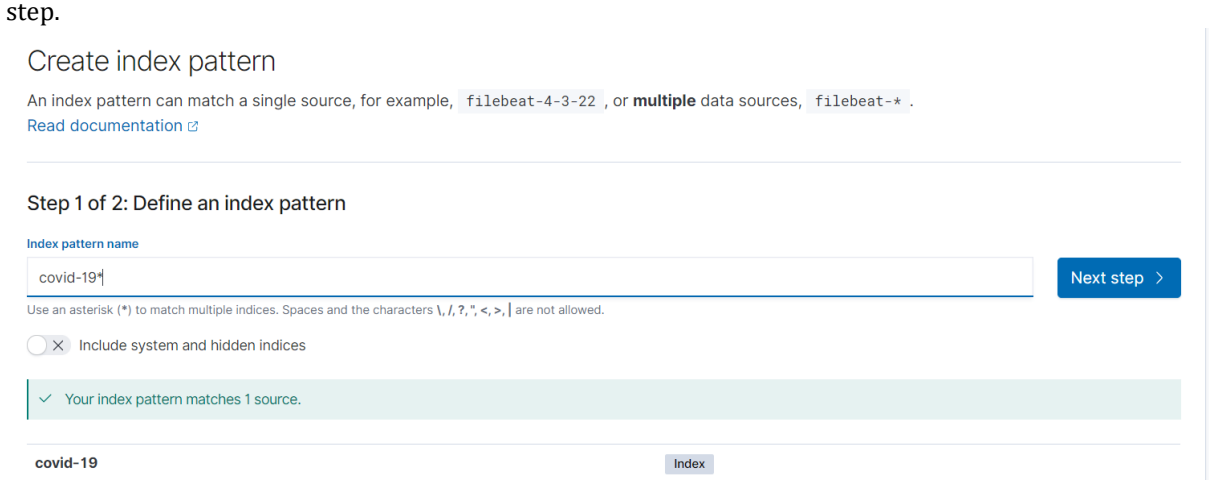

- Now, we can check for indexed documents in Kibana. Go to http://localhost:5601/, search for index patterns. In the figure below, we can see that **covid-19** index is there. This is the name given to index in logstash.conf file.



- We can see the **covid-19** index has 1000 documents.

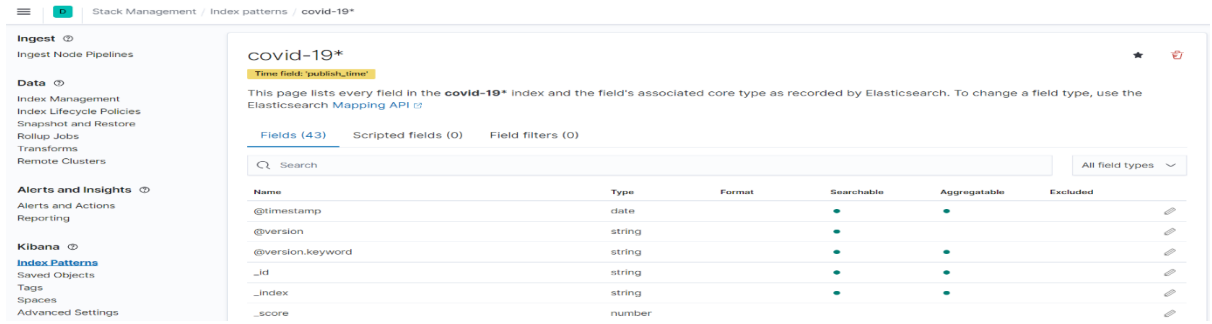


- **Creating index pattern**: In step1 define index pattern, **covid-19** is the index so type covid-19, this pops up a message **'your index pattern matches 1 source covid-19'**. Then click on next step.



- In step2, configure the time field. I choose publish_time as the time field and click next.

- This finishes index pattern. Every filed is indexed with covid-19 index.



- A simple query to obtain all the titles and journal names with covid-19 index looks like this:

- Go to discover section in Kibana and save the indexed file as .csv file. The indexed file looks like this:

```
pd.read_csv("CORD_19_indexed.csv").head(3)
```

	@timestamp	@version	_id	_index	_score	_type	abstract	authors	cord_uid	doi	...	
0	2021-03-02T00:37:43.421Z	1	4Mhe8HcBhQmyCbDQi6-I	covid-19	NaN	_doc	The most severe manifestations of malaria (cau...	Conant, Katelyn L.; Kaleeba, Johnan A. R.	d0bk9gu5	10.3389/fmicb.2013.00035	...	C:/Users/j Rereiev
1	2021-03-02T00:37:43.421Z	1	z8he8HcBhQmyCbDQjrLY	covid-19	NaN	_doc	NOA36/ZNF330 is an evolutionarily well-preserv...	de Melo, Ivan S.; Jimenez-Nuñez, Maria D.; Igl...	0jx6mwiw	10.1371/journal.pone.0059065	...	C:/Users/j Rereiev
2	2021-03-02T00:37:43.420Z	1	V8he8HcBhQmyCbDQi7GN	covid-19	NaN	_doc	Global climate change is expected to affect th...	CANN, K. F.; THOMAS, D. Rh.; SALMON, R. L.; WY...	exqza1kg	10.1017/s0950268812001653	...	C:/Users/j Rereiev

3. Sentence Splitting, Tokenization and Normalization:

- From the 'docs' data frame, the 'title' and 'abstract' columns are chosen for preprocessing. The 'title' and 'abstract' contain important information about an article; hence these two columns are chosen for preprocessing and later to select keywords.
- The title and abstract columns are concatenated into a single column called 'text'. The following steps are followed in preprocessing:

1. First all the irrelevant characters (numbers and punctuations) are removed
2. Second, all characters are converted into lowercase
3. Thirdly, the sentences are converted into tokens (Tokenization)
4. After that, all English the stop words are removed
5. Then, Lemmatization is performed on tokens to extract correct base forms of words.
6. Any tokens/words having length <=2 is removed
7. Finally, the tokens are joined into string

Function for data preprocessing

```
lemma= WordNetLemmatizer() # initializing object of WordNetLemmatizer for lemmatization
stop_words = set(stopwords.words('english')) # taking all the unique English stopwords from nltk corpus

def preprocess(text):
    text = re.sub('[^a-zA-Z]',' ',text) # removing numbers and punctuations
    text = str(text).lower()            # convert all characters into lowercase
    text = word_tokenize(text)          # tokenization
    text = [item for item in text if item not in stop_words] # removing stopwords
    text = [lemma.lemmatize(word=w,pos='v') for w in text]   # lemmatization
    text = [i for i in text if len(i) > 2]   # removing token of length <=2
    text = ' '.join(text)                    # joining the tokens with space in between to form sentence

    return text

docs['text'] = docs['title'] + docs['abstract'] # combining the title and abstract column into a single column called text
docs['text'] = docs['text'].apply(lambda x: preprocess(x)) # applying the 'preprocess' function on 'text' column
docs['text'][0] # visualizing the processed text
```

```
'clinical feature culture prove mycoplasma pneumoniae infections king abdulaziz university hospital jeddah saudi arabiaobjectiv
e retrospective chart review describe epidemiology clinical feature patients culture prove mycoplasma pneumoniae infections kin
g abdulaziz university hospital jeddah saudi arabia methods patients positive pneumoniae culture respiratory specimens january
december identify microbiology record chart patients review result patients identify require admission infections community acq
uire infection affect age group common infants pre school children occur year round common fall spring three quarter patients c
omorbidities twenty four isolate associate pneumonia upper respiratory tract infections bronchiolitis cough fever malaise commo
n symptoms crepitations wheeze common sign patients pneumonia crepitations bronchial breathe immunocompromised patients likely
non immunocompromised patients present pneumonia versus patients pneumonia uneventful recovery recover follow complications die
pneumoniae infection die due underlie comorbidities patients die pneumoniae pneumonia comorbidities conclusion result similar p
ublish data except find infections common infants preschool children mortality rate pneumonia patients comorbidities high'
```

- The figures below show the comparison before and after the preprocessing:
  For single document, nearly 700 words are removed after preprocessing, all the characters are converted into lower case, numbers, symbols are eliminated, only meaningful words are retained.

```
text_before_preprocess = covid_df.title[0] + covid_df.abstract[0]
print("Text before preprocessing : ","\n")
print(text_before_preprocess,'\n')
print("length of text before preprocessing : ",len(text_before_preprocess))
```

```
Text before preprocessing :

Clinical features of culture-proven Mycoplasma pneumoniae infections at King Abdulaziz University Hospital, Jeddah, Saudi Arabi
aOBJECTIVE: This retrospective chart review describes the epidemiology and clinical features of 40 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 1997 through December 1998 were identified through the Microbiol
ogy records. Charts of patients were reviewed. RESULTS: 40 patients were identified, 33 (82.5%) of whom required admission. Mos
t infections (92.5%) were community-acquired. The infection affected all age groups but was most common in infants (32.5%) and
pre-school children (22.5%). It occurred year-round but was most common in the fall (35%) and spring (30%). More than three-qua
rters of patients (77.5%) had comorbidities. Twenty-four isolates (60%) were associated with pneumonia, 14 (35%) with upper res
piratory tract infections, and 2 (5%) with bronchiolitis. Cough (82.5%), fever (75%), and malaise (58.8%) were the most common
symptoms, and crepitations (60%), and wheezes (40%) were the most common signs. Most patients with pneumonia had crepitations
(79.2%) but only 25% had bronchial breathing. Immunocompromised patients were more likely than non-immunocompromised patients t
o present with pneumonia (8/9 versus 16/31, P = 0.05). Of the 24 patients with pneumonia, 14 (58.3%) had uneventful recovery, 4
(16.7%) recovered following some complications, 3 (12.5%) died because of M pneumoniae infection, and 3 (12.5%) died due to und
erlying comorbidities. The 3 patients who died of M pneumoniae pneumonia had other comorbidities. CONCLUSION: our results were
similar to published data except for the finding that infections were more common in infants and preschool children and that th
e mortality rate of pneumonia in patients with comorbidities was high.

length of text before preprocessing :  1975
```

```
print("Text after preprocessing : ","\n")
print(docs['text'][0],'\n')
print("length of text after preprocessing : ",len(docs['text'][0]))
```

```
Text after preprocessing :

clinical feature culture prove mycoplasma pneumoniae infections king abdulaziz university hospital jeddah saudi arabiaobjective
retrospective chart review describe epidemiology clinical feature patients culture prove mycoplasma pneumoniae infections king
abdulaziz university hospital jeddah saudi arabia methods patients positive pneumoniae culture respiratory specimens january de
cember identify microbiology record chart patients review result patients identify require admission infections community acqui
re infection affect age group common infants pre school children occur year round common fall spring three quarter patients com
orbidities twenty four isolate associate pneumonia upper respiratory tract infections bronchiolitis cough fever malaise common
symptoms crepitations wheeze common sign patients pneumonia crepitations bronchial breathe immunocompromised patients likely no
n immunocompromised patients present pneumonia versus patients pneumonia uneventful recovery recover follow complications die p
neumoniae infection die due underlie comorbidities patients die pneumoniae pneumonia comorbidities conclusion result similar pu
blish data except find infections common infants preschool children mortality rate pneumonia patients comorbidities high

length of text after preprocessing :  1264
```

4. Stemming or Morphological Analysis

- Initially, I performed **stemming** to normalize the data, but I found that the words generated have no clear meaning. The figure below displays some words created after stemming and we can clearly see they don't have any meaning; they were just chopped off into base form.

```
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

def stemming(token):
    return [stemmer.stem(w) for w in token]

for column in columns_for_prpcessing.columns:
    columns_for_prpcessing[column] = columns_for_prpcessing[column].apply(stemming)
```

```
columns_for_prpcessing[column][0]
```

```
['object',
 'retrospect',
 'chart',
 'review',
 'describ',
 'epidemiolog',
 'clinic',
 'featur',
 'patient',
 'cultur',
 'prove',
 'mycoplasma',
 'pneumonia',
 'infect',
 'king',
 'abdulaziz',
 'univ',
 'hospit',
 'jeddah',
```

- **In the 'preprocess' function I used lemmatization to normalize the text and the tokens created have clear meaning to them. We can clearly see the difference between stemming and lemmatization in words highlighted. So, I choose to perform lemmatization.**

- For instance, the university is chopped off into ‘univ’, ‘hospital’ into ‘hospit’ using stemming. These words don’t have any meaning and if we apply stemming as normalization, then later these words are used to extract key words, and this will negatively impact the search.

5. Selecting Keywords:

- The pre-processed text column that is ‘docs[‘text’]’ column is used to extract keywords in each document. This process involves 2 stages.
- **Step1:**

Building Vocabulary and selecting keywords

```
documents = docs['text'].tolist() # getting the text column and converting it into a list
```

```
# creating a vocabulary of words, ignoring the word that appear in 85% of documents
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_df=0.85)
token_count_vector = vectorizer.fit_transform(documents)
# creates the vocabulary, the result is a sparse representation of count of each word
```

```
token_count_vector.shape
# The shape is (1000,12295) since we have 1000 documents and vocabulary size is 12295
```

```
(1000, 12295)
```

```
token_count_vector.toarray()[0]
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
list(vectorizer.vocabulary_.keys())[:5] # looking at first 5 words of vocabulary
```

```
['clinical', 'feature', 'culture', 'prove', 'mycoplasma']
```

CountVectorizer from sklearn library is used to create vocabulary. CountVectorizer converts a collection of text documents to a matrix of token counts, for CountVectorizer we passed an argument ‘max_df=0.85’, this argument ignores the words that appear more than 85% times in a document.
For 1000 documents, a vocabulary of 12,295 words is created.

- **Step2:**

Extracting keywords

```
# First compute the IDF(InverseDocumentFrequency) values. For this, we take  the sparse matrix generated from CountVectorizer
# (token_count_vector) to calculate the IDF by invoking tfidf_transformer.fit(...)

tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
    tfidf_transformer.fit(word_count_vector)
```

```
feature_names=vectorizer.get_feature_names() # get all the words from vocabulary

doc = docs['text'] # This is the column for which we want to generate tf-idf
results = [] # list to store the extracted keywords for each document
for d in doc:
     #Transform a count matrix to a normalized tf or tf-idf representation
    tf_idf_vector = tfidf_transformer.transform(vectorizer.transform([d]))

    #sorting the tf-idf vectors by descending order of scores by calling the function "sort_tf_idf_vector()"
    sorted_items=sort_tf_idf_vector(tf_idf_vector.tocoo())

    #extract only the top n; n here is top 20% words with high tf-idf values by calling the function "extract_topn_from_vector()"
    keywords=extract_topn_from_vector(feature_names,sorted_items,int(len(sorted_items)*0.2))

    results.append(' '.join(keywords))
```

```
def sort_tf_idf_vector(coo_matrix):
    """Sorting the tf-idf vector"""
    tuples = zip(coo_matrix.col, coo_matrix.data)
    return sorted(tuples, key=lambda x: (x[1], x[0]), reverse=True)

def extract_topn_from_vector(feature_names, sorted_items,topn):
    """get the feature names and tf-idf score of top 20% words"""

    sorted_items = sorted_items[:topn]  #collect only topn items from vector

    topnwords = []

    # index of word, respective tf-idf score
    for idx, score in sorted_items:

        #append the sorted words
        topnwords.append(feature_names[idx])

    return topnwords
```

```
results
```

```
 'nef partner interaction membrane screen hiv hereby gpm identify proteins hybrid protein nucleus yeast interactions vsig tsp
an tcblr subfraction proteinshiv pmepa',
 'pertussis household infants korea infant vaccination transmission young months immunization source contact tdap lymphocytos
is koreaa dtap resurgence booster age',
 'lofreq sequence variants datasets population heterogeneity rare call throughput bacterial exist sourceforge sequenom gastri
c fluidigm exome executables datasetsthe caller',
 'buho foci sgs pbs stress matlab image script throughput repress formation process manually granules aggregate silence autom
ate mrnas cellular reliable body experiment mammalian proteins mrna upon analysis term',
 'spp sargassum polysaccharides pallidum newcastle bronchitis adjuvant chickens lymphocyte inactivate',
 'india inequalities disparities pandemic health source influenza plausible declare level geographic social access preparedne
ss plan factor world model unequal',
```

- First compute Inverse Document Frequency (IDF) using TfidfTransformer. Then, we compute the tf-idf value for each document by invoking tfidf_transformer. transform(). This generates a vector of tf-idf scores. After this, we sort the words in the vector in descending order of tf-idf scores and then iterate over to extract the top-n key words.

- Now, we extracted top 20% keywords for each document. These keywords are concatenated as new column to the 'docs' data frame and this data frame is reindexed.

```
docs['keywords'] = results
```

```
docs.head(3)
```

pdf_json_files	pmc_json_files	url	text	keywords
nt_parses/pdf_json/d1aafb70c066a2068b027...	document_parses/pmc_json/PMC35282.xml.json	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3...	clinical feature culture prove mycoplasma pneu...	pneumoniae patients comorbidities pneumonia co...
t_parses/pdf_json/6b0567729c2143a66d737...	document_parses/pmc_json/PMC59543.xml.json	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5...	nitric oxide pro inflammatory mediator lung di...	inflammatory oxidant presume nitric oxide cont...

- This data frame is saved to a csv file for indexing.

```
docs.drop(columns='text',inplace=True)
```

```
docs.columns
```

```
Index(['cord_uid', 'sha', 'source_x', 'title', 'doi', 'pmcid', 'pubmed_id',
       'license', 'abstract', 'publish_time', 'authors', 'journal',
       'pdf_json_files', 'pmc_json_files', 'url', 'keywords'],
      dtype='object')
```

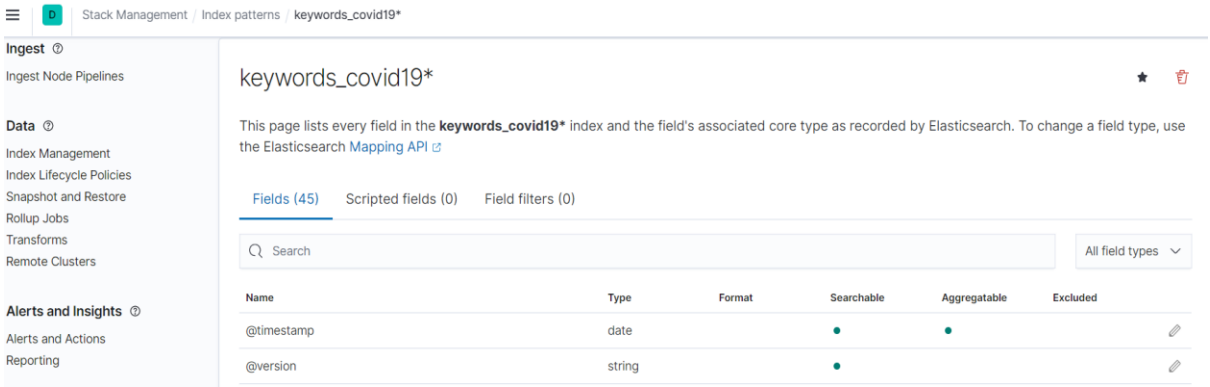
```
docs.to_csv("keywords_added_docs.csv",index=False)
```

6. Reindexing:

- A new Logstash configuration file is created to index this "keywords_added_docs.csv" file.
- The following figure displays the contents of 'newlogstash.conf'.

```
input {
  file {
      path =>
"C:/Users/jhans/OneDrive/Documents/keywords_added_docs.csv"
      start_position => "beginning"
      sincedb_path => "NULL"
      }
}
filter {
  csv{
      separator => ","
      columns => ["cord_uid", "sha", "source_x", "title", "doi",
"pmcid", "pubmed_id","license", "abstract","publish_time",
"authors", "journal", "pdf_json_files",
"pmc_json_files","url","keywords"]
      }
}
output {
  elasticsearch
      {
      hosts => ["http://localhost:9200"]
      index => "keywords_covid19"
      }
      stdout {}
}
```

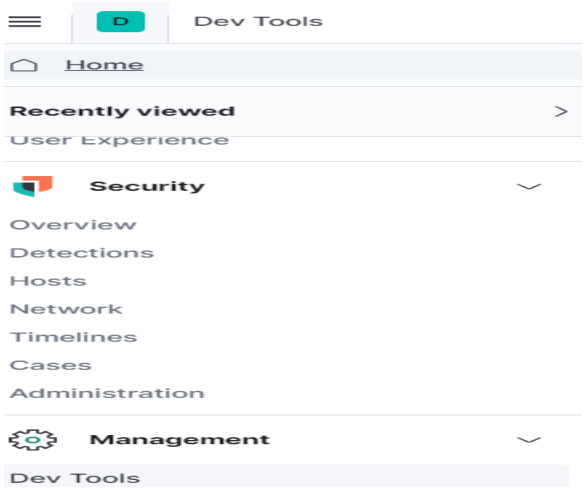
- **All the 1000 documents are now indexed with an index "keywords_covid19".**
- Now, we can check for indexed documents in Kibana. Go to http://localhost:5601/, search for index patterns. In the figure below, we can see that **keywords_covid19** index is there. This is the name given to index in newlogstash.conf file.



- We can now query these indexed documents in Elasticsearch.

7. Searching:

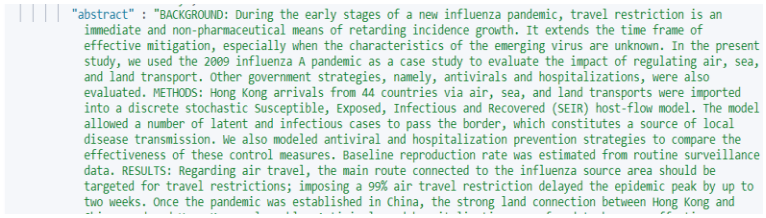
We can now write queries and search the indexed documents in Kibana. Go to http://localhost:5601/. In Kibana select 'Devtools' under 'management' section. We can write queries here and retrieve relevant documents.

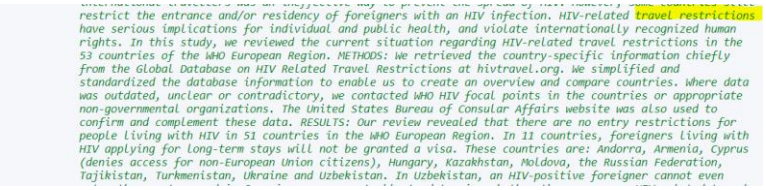


For the following phrases the queries are formed:

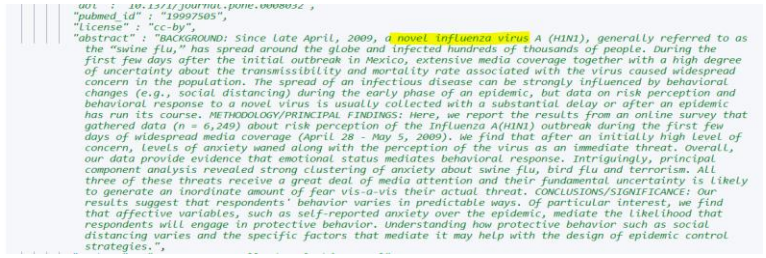
1. Travel restrictions
2. Novel corona virus
3. Antiviral treatment
4. Transmission
5. Preventing transmission
6. Seasonality of transmission
7. Retrieving articles based on author name
8. Retrieving articles based on publishing time
9. Vaccination
10. Obtaining articles based on cord_uid

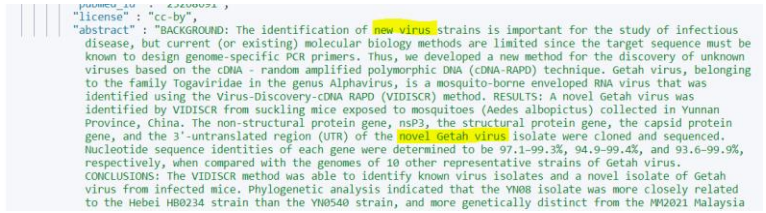
1.Travel restrictions:



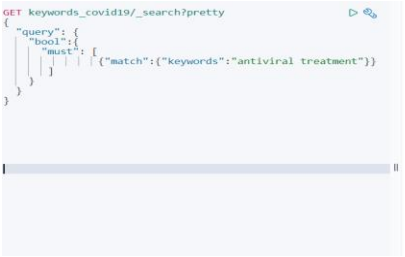


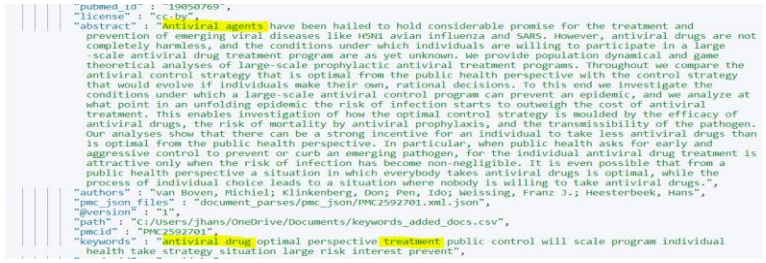
2. Novel viruses:





3. Antiviral Treatment:





```
GET keywords_covid19/_search?pretty
{
  "query": {
    "bool": {
      "must":[
        {"match":{"title":"novel"}},
        {"match":{"keywords":"antiviral treatment"}}

      ]
    }
  }
}
```

```
"keywords" : "tmv phenanthroquinolizidine alkaloids configuration activity antiviral tobacco mosaic
  introduction compound vivo stucture phenanthroquinolizidines",
"cord_uid" : "5is9kc52",
"source_x" : "PMC",
"@timestamp" : "2021-03-03T19:39:46.923Z",
"host" : "LAPTOP-BHI84IJF",
"pdf_json_files" : "document_parses/pdf_json/a4ffcadecc4b60c30df8f699c480724523272e62.json",
"title" : "First Discovery and Stucture-Activity Relationship Study of Phenanthroquinolizidines as Novel
  Antiviral Agents against Tobacco Mosaic Virus (TMV)",
"url" : "https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3532156/",
"journal" : "PLoS One",
```

4. Transmission of disease:

```
GET keywords_covid19/_search?pretty
{
  "query": {
    "bool":{
      "must": [
        {"match":{"keywords":"transmission"}}
      ]
    }
  }
}
```

```
"publish_time" : "2010-12-28",
"doi" : "10.1155/2011/652652",
"pubmed_id" : "23074659",
"license" : "cc-by",
"abstract" : "The 2003-2004 H5N1 highly pathogenic avian influenza (HPAI) outbreaks in Japan were the first
  such outbreaks in 79 years in Japan. Epidemic outbreaks have been occurring in Southeast Asia, with the
  most recent in 2010. Knowledge of the transmission route responsible for the HPAI outbreaks in these
  countries remains elusive. Our studies strongly suggested that field and laboratory studies focusing on
  mechanical transmission by blow flies should be considered to control H5N1 avian influenza outbreaks, in
  particular in epidemic areas, where there are high densities of different fly species throughout the year.
  In this paper, we review these field and laboratory entomological studies and discuss the possibility of
  blow flies transmitting H5N1 viruses.",
"authors" : "Sawabe, Kyoko; Hoshino, Keita; Isawa, Haruhiko; Sasaki, Toshinori; Kim, Kyeong Soon; Hayashi,
  Toshihiko; Tsuda, Yoshio; Kurahashi, Hiromu; Kobayashi, Mutsuo",
"pmc_json_files" : "document_parses/pmc_json/PMC3447300.xml.json",
"@version" : "1",
"path" : "C:/Users/jhans/OneDrive/Documents/keywords_added_docs.csv",
"pmcid" : "PMC3447300",
"keywords" : "fly outbreaks blow japan avian hpai transmission pathogenic field epidemic",
"cord_uid" : "jx518gz3",
"source_x" : "PMC",
"@timestamp" : "2021-03-03T19:39:46.907Z",
"host" : "LAPTOP-BHI84IJF",
"pdf_json_files" : "document_parses/pdf_json/47939863a50677044e74f69efc9b22a80d37562f.json",
"title" : "Blow Flies Were One of the Possible Candidates for Transmission of Highly Pathogenic H5N1 Avian
  Influenza Virus during the 2004 Outbreaks in Japan",
```

5. Preventing transmission:

```
GET keywords_covid19/_search?pretty
{
  "query": {
    "match":{
      "keywords": {
        "query": "prevention transmission",
        "operator" : "and",
        "minimum_should_match": "75%"
      }
    }
  }
}
```

```
"sha" : "8ffdabbd3e504f20d074aa3c41a09aa85b855998",
"publish_time" : "2009-05-13",
"doi" : "10.1186/1478-4491-7-39",
"pubmed_id" : "19439094",
"license" : "cc-by",
"abstract" : "BACKGROUND: Prevention of mother-to-child transmission has been considered as not a simple
  intervention but a comprehensive set of interventions requiring capable health workers. Viet Nam's
  extensive health care system reaches the village level, but still HIV-infected mothers and children have
  received inadequate health care services for prevention of mother-to-child transmission. We report here the
  health workers' perceptions on factors that lead to their failure to give good quality prevention of mother
  -to-child transmission services. METHODS: Semistructured interviews with 53 health workers and unstructured
  observations in nine health facilities in Hanoi were conducted. Selection of respondents was based on their
  function, position and experience in the development or implementation of prevention of mother-to-child
  transmission policies/programmes. RESULTS: Factors that lead to health workers' failure to give good
```

6. Seasonality of transmission:

```
GET keywords_covid19/_search?pretty
{
  "query": {
    "match":{
      "keywords": {
        "query": "season transmission",
        "operator" : "and",
        "minimum_should_match": "75%"
      }
    }
  }
}
```

```
"_score" : 8.92118,
"_source" : {
  "sha" : "ec84301b786a7e9bb2745f5a4141a69e484a92f3",
  "publish_time" : "2009-01-04",
  "doi" : "10.1155/2009/591935",
  "pubmed_id" : "19266090",
  "license" : "cc-by",
  "abstract" : "Seasonal variation in smallpox transmission is one of the most pressing ecological questions
    and is relevant to bioterrorism preparedness. The present study reanalyzed 7 historical datasets which
    recorded monthly cases or deaths. In addition to time series analyses of reported data, an estimation and
    spectral analysis of the effective reproduction number at calendar time t, R(t), were made. Meteorological
    variables were extracted from a report in India from 1890–1921 and compared with smallpox mortality as well
    as R(t). Annual cycles of smallpox transmission were clearly shown not only in monthly reports but also in
    the estimates of R(t). Even short-term epidemic data clearly exhibited an annual peak every January. Both
    mortality and R(t) revealed significant negative association (P < .01) and correlation (P < .01),
    respectively, with humidity. These findings suggest that smallpox transmission greatly varies with season
    and is most likely enhanced by dry weather.",
```

7. Retrieving articles based on author name:

```
GET keywords_covid19/_search?pretty
{
  "query": {
    "term" : {
      "authors": "aisha"
    }
  },
  "_source": ["title","authors"]
}
```

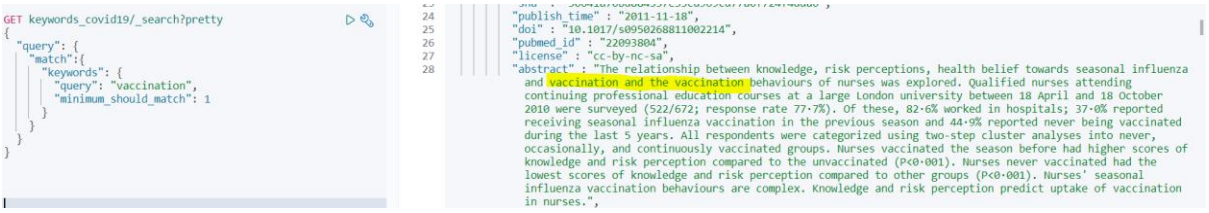
```
{
  "took" : 0,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 8.474398,
    "hits" : [
      {
        "_index" : "keywords_covid19",
        "_type" : "_doc",
        "_id" : "Lq2a-XcBfaEaBUoEgNyB",
        "_score" : 8.474398,
        "_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"
        }
      }
    ]
  }
}
```

8. Retrieving articles based on publishing time

```
GET keywords_covid19/_search?pretty
{
  "query": {
    "range" : {
      "publish_time": {
        "gte":"2006-04-25",
        "lte":"2020-11-26"
      }
    }
  },
  "_source" : ["title","publish_time","journal"],
  "sort": [
    { "publish_time": {"order":"desc"}}
  ]
}
```

```
"hits" : [
  {
    "_index" : "keywords_covid19",
    "_type" : "_doc",
    "_id" : "rK2a-XcBfaEa8UoEgNuB",
    "_score" : null,
    "_source" : {
      "journal" : "Curr Genomics",
      "publish_time" : "2013-03-22",
      "title" : "Synthetic Genomics and Synthetic Biology Applications Between Hopes and Concerns"
    },
    "sort" : [
      1363910400000
    ]
  },
  {
    "_index" : "keywords_covid19",
    "_type" : "_doc",
    "_id" : "IK2a-XcBfaEa8UoEh95l",
    "_score" : null,
    "_source" : {
      "journal" : "PLoS One",
      "publish_time" : "2013-03-13",
      "title" : "NOA36 Protein Contains a Highly Conserved Nucleolar Localization Signal Capable of Directing
       Functional Proteins to the Nucleolus, in Mammalian Cells"
    },
    "sort" : [
      1363132800000
    ]
  },
```

9. Vaccination



```
GET keywords_covid19/_search?pretty
{
  "query": {
    "match":{
      "keywords": {
        "query": "vaccination",
        "minimum_should_match": 1
      }
    }
  }
}
```

```
"publish_time" : "2011-11-18",
"doi" : "10.1017/s0950268811002214",
"pubmed_id" : "22093804",
"license" : "cc-by-nc-sa",
"abstract" : "The relationship between knowledge, risk perceptions, health belief towards seasonal influenza
 and vaccination and the vaccination behaviours of nurses was explored. Qualified nurses attending
 continuing professional education courses at a large London university between 18 April and 18 October
 2010 were surveyed (522/672; response rate 77·7%). Of these, 82·6% worked in hospitals; 37·0% reported
 receiving seasonal influenza vaccination in the previous season and 44·9% reported never being vaccinated
 during the last 5 years. All respondents were categorized using two-step cluster analyses into never,
 occasionally, and continuously vaccinated groups. Nurses vaccinated the season before had higher scores of
 knowledge and risk perception compared to the unvaccinated (P<0·001). Nurses never vaccinated had the
 lowest scores of knowledge and risk perception compared to other groups (P<0·001). Nurses' seasonal
 influenza vaccination behaviours are complex. Knowledge and risk perception predict uptake of vaccination
 in nurses.",
```

```
GET keywords_covid19/_search?pretty
{
  "query": {
    "match": {
      "keywords":{
        "query": "vaccines pandemic",
        "operator":"and"}
    }
  }
}
```

```
{
  "_index" : "keywords_covid19",
  "_type" : "_doc",
  "_id" : "uq2a-XcBfaEa8UoEgNqB",
  "_score" : 7.339862,
  "_source" : {
    "sha" : "56e040a2052581a7133f24e50dc7261b579fe22f",
    "publish_time" : "2007-10-22",
    "doi" : "10.3201/eid1310.061262",
    "pubmed_id" : "18258000",
    "license" : "no-cc",
    "abstract" : "Influenza A (H5N1) viruses are strong candidates for causing the next influenza pandemic if
     they acquire the ability for efficient human-to-human transmission. A major public health goal is to make
     efficacious vaccines against these viruses by using novel approaches, including cell-culture system,
     reverse genetics, and adjuvant development. Important consideration for the strategy includes preparation
     of vaccines from a currently circulating strain to induce broad-spectrum immunity toward newly emerged
     human H5 strains. This strategy would be a good solution early in a pandemic until an antigenically
     matched and approved vaccine is produced. The concept of therapeutic vaccines (e.g., antidisease vaccine)
     directed at diminishing the cytokine storm frequently seen in subtype H5N1–infected persons is underscore
     . Better understanding of host–virus interaction is essential to identify tools to produce effective
     vaccines against influenza (H5N1).",
```

10. Obtaining articles based on cord_uid

```
GET keywords_covid19/_search?pretty
{
  "query": {
    "term": {
      "cord_uid":"02tnwd4m"
    }
  }
}
```

```
{
  "_index" : "keywords_covid19",
  "_type" : "_doc",
  "_id" : "OK2a-XcBfaEa8UoEgNqA",
  "_score" : 6.5032897,
  "_source" : {
    "sha" : "6b0567729c2143a66d737eb0a2f63f2dce2e5a7d",
    "publish_time" : "2000-08-15",
    "doi" : "10.1186/rr14",
    "pubmed_id" : "11667967",
    "license" : "no-cc",
    "abstract" : "Inflammatory diseases of the respiratory tract are commonly associated with elevated production
     of nitric oxide (NO•) and increased indices of NO• -dependent oxidative stress. Although NO• is known to
     have anti-microbial, anti-inflammatory and anti-oxidant properties, various lines of evidence support the
     contribution of NO• to lung injury in several disease models. On the basis of biochemical evidence, it is
     often presumed that such NO• -dependent oxidations are due to the formation of the oxidant peroxynitrite,
     although alternative mechanisms involving the phagocyte-derived heme proteins myeloperoxidase and
     eosinophil peroxidase might be operative during conditions of inflammation. Because of the overwhelming
     literature on NO• generation and activities in the respiratory tract, it would be beyond the scope of this
     commentary to review this area comprehensively. Instead, it focuses on recent evidence and concepts of the
     presumed contribution of NO• to inflammatory diseases of the lung.",
    "authors" : "Vliet, Albert van der; Eiserich, Jason P; Cross, Carroll E",
    "pmc_json_files" : "document_parses/pmc_json/PMC59543.xml.json",
    "@version" : "1",
    "path" : "C:/Users/jhans/OneDrive/Documents/keywords_added_docs.csv",
    "pmcid" : "PMC59543",
    "keywords" : "inflammatory oxidant presume nitric oxide contribution lung anti evidence tract phagocyte
     peroxynitrite oxidations comprehensively",
    "cord_uid" : "02tnwd4m",
    "source_x" : "PMC",
```