**TOPIC MODELING**

**DISSERTATION**

*Submitted in partial fulfilment of the*

*Requirements for the award of the degree*

*Of*

**Bachelor of Technology**

in

**Information Technology & Engineering**

By:

**Arpit Singh (417/IT3/2018)**

****

**Department of Information Technology & Engineering**

**Guru Tegh Bahadur Institute of Technology**

**Guru Gobind Singh Indraprastha University**

**Dwarka, New Delhi**

**Year 2018-2022**

**DECLARATION**

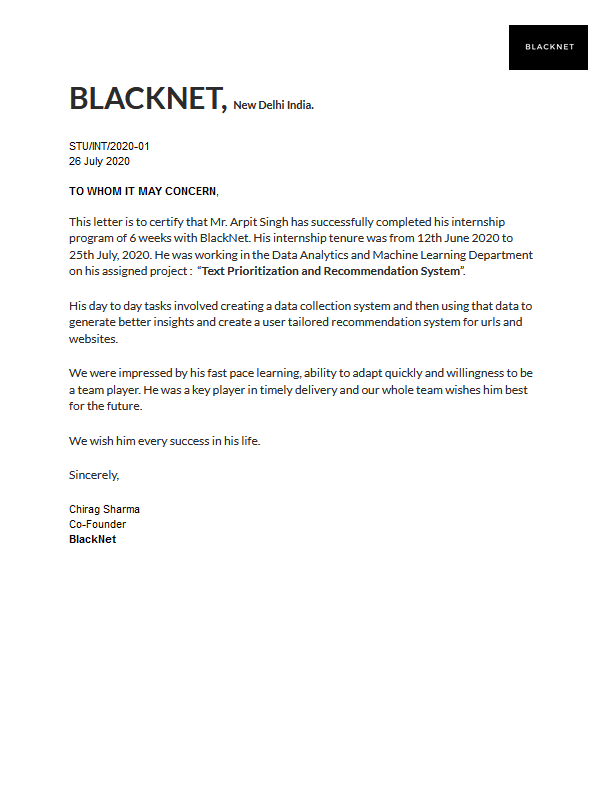
I hereby declare that all the work presented in this Industrial Training Report for the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering, Guru Tegh Bahadur Institute of Technology, affiliated to Guru Gobind Singh Indraprastha University Delhi is an authentic record of our own work carried out at Microsoft Technologies, Redmond, Washington, United States from 7 th May, 2020 to 30th June, 2020.

**Date: 1-12-2020**

**Arpit Singh (417/IT3/2018)**

**ii**

**CERTIFICATE**

****

**iii**

**ACKNOWLEDGEMENT**

I would like to express our great gratitude towards Mr. GAURAV SANDHU who has given me support and this opportunity to explore new technologies. Without his help I could not have presented this work up to the present standard. I also take this opportunity to give thanks to all others who gave me support for the project or in other aspects of our study at Guru Tegh Bahadur Institute of Technology

**Arpit Singh (417/IT3/2018)**

**(** [charstring944@gmail.com](mailto:charstring944@gmail.com) **)**

**Date:1-12-2020**

**iv**

**ABSTRACT**

NLP is a part of UNSUPERVISED machine learning with the ability of a computer to understand, analyse, manipulate, and potentially generate human language.

In this project I was allotted the task of analysing the data and finding the topics that are most widely spread and also the one that are least or underrated in the year 2020. The dataset provided contains the data of about 8 months (NOVEMBER 2019 – JUNE 2020) which contains the KEYWORDS, HASTAGS, HEADINGS, SUMMARY, CITY, STATE and many other features from newspaper, articles and magazines. I used technique of NALTURAL LANGUAGE PROCESSING with libraries like NLTK (is a popular open-source package in Python. Rather than building all tools from scratch, NLTK provides all common NLP Tasks.), Word2Vec, LDA and Word Cloud to analyse the data and come up with the most and least frequent keywords and find the topics that were used between November 2019–June 2020, so that they can use the useful insights and made changes in the website to rank the most useful news to get the traffic and cover those topics also that are underrated and not given importance due to COVID-19.

Companies are increasingly using NLP-equipped tools to gain insights from data and to automate routine tasks. Probably, the most popular examples of NLP in action are virtual assistants, like Google Assist, Siri, and Alexa. NLP understands and translates the human language, like “Hey Siri, where is the nearest gas station?” into numbers, making it easy for machines to understand. NLP tools are helping companies understand how their customers perceive them across all channels of communication, whether emails, product reviews, social media posts, surveys, and more. The insight was all about the CORONAVIRUS and the harmful effects on various sectors like government, banks, poor which leads to the result of lockdown and LDA (Latent Dirichlet Allocation) works the best as it took less time and runs on CPU so we don’t need to worry about the GPU and cores in the laptop.



**v**

**TABLE OF CONTENTS**

**CHAPTER PAGE NO.**

|  |
| --- |
| Title Page i |
| Declaration ii |
| Certificate iii |
| Acknowledgement iv |
| Abstract v |
| Table of Contents vi-vii |
| List of Figures viii |
| 1. INTRODUCTION |
| * 1. About Summer Training 1 |
| * 1. Company Profile 1 |
| * 1. Technology 2 |
| * 1. Application 2 |
| * 1. Challenges Faced 3 |
| * 1. Dataset 3 |
| * 1. Objective 4 |
| 1. TECHNOLOGY |
| 2.1 Python 6 |
| 2.2 NLP (Natural Language Processing) 7 |
| 2.3 IDE 18 |

**vi**

**CHAPTER PAGE NO.**

|  |
| --- |
| 3. DATA CLEANING and ANALYSING |
| 3.1 Cleaning the data 20 |
| 3.2 Word2Vec 22 |
| 3.3 Latent Dirichlet Allocation (LDA) 24 |
| 3.4 Visual Analysis 25 |
| 4. RESULT & OBSERVATION 27 |
| 5. CONCLUSION & FUTURE SCOPE 30 |
| 6. REFERENCE 31 |
| 7. APPENDIX |
| 7.1 Screenshot 34 |
| 7.2 Source Code 39 |

**vii**

**LIST OF FIGURES**

**FIGURE PAGE**

|  |
| --- |
| Fig.1 2 |
| Fig.2 4 |
| Fig.3 6 |
| Fig.4 7 |
| Fig.5 8 |
| Fig.6 9 |
| Fig.7 18 |
| Fig.8 20 |
| Fig.9 22 |
| Fig.10 23 |
| Fig.11 24 |
| Fig.12 25 |
| Fig.13 27 |
| Fig.14 28 |
| Fig.15 34 |
| Fig.16 35 |
| Fig.17 36 |
| Fig.18 37 |

**viii**

**CHAPTER**

**ONE**

**INTRODUCTION**

**1.1 ABOUT SUMMER TRAINING**

My summer training had started on **12 June 2020** under the **Blacknet** startup. It was a training of approximately 6 weeks. My training was based on NLP (Natural Language Processing), I was selected on my previous experience and fundamentals of Data Science and Machine Learning. I was working as a DATA ANALYST on the project: **“TEXT PRIORITIZATION AND RECOMMENDATION SYSTEM”**.

My day to day tasks involved creating a data collection system, clearing the data, making the data ready in the readable and easy understandable form and then using that data to generate better insights and create a user tailored system for the URLs and website.

I was completely new towards the NLP field and was much into Machine Learning and Data Science. But it was a really good experience to explore the new field of AI as NLP is one of the booming technologies that is spreading like fire. I learned a lot in these 6 weeks and hope to be the part of the same industry as I’m also currently working in this startup as Web Developer also.

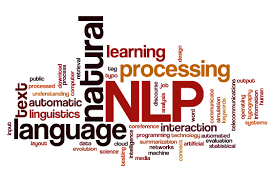
**1.2 COMPANY PROFILE**

BlackNet is Delhi Based AI Startup whose main focus is to create AI Solutions for other startups, companies other startups and small tech business to integrate Machine Learning, Deep Learning and AI services in their work flows That help clients to discover data potential, AI and ML use cases to generate tangible profits and better revenues.

It has been working on challenging problems for last 2+ years and Worked with multiple clients now. Due to their cross-domain experience and expertise they are able to deliver multiple projects across the countries. BlackNet has worked on multiple projects and served 5+ client based both national and international.

**1**

**1.3 TECHNOLOGY**



Computers and machines are great at working with tabular data or spreadsheets. However, as human beings generally communicate in words and sentences, not in the form of tables. Much information that humans speak or write is unstructured. So it is not very clear for computers to interpret such. In natural language processing (NLP), the goal is to make computers understand the unstructured text and retrieve meaningful pieces of information from it. Natural language Processing (NLP) is a subfield of *ARTIFICIAL INTELLIGENCE*, in which its depth involves the interactions between computers and humans.

**1.4 APPLICATION**

**. Intelligent document analysis**

Combine NLP and machine learning (ML) to help gain insights into human-generated, natural language text documents.

**. Business chatbots and virtual assistants**

Answer support queries and direct users to manuals or other resources, helping enterprises reduce support costs and improve customer engagement.

**. Sentiment analysis and market intelligence**

Mine social media, reviews, news, and other relevant sources to gain better insights about customers, partners, competitors, and market trends.

**. Fraud detection and risk management**

Identify potential fraud and risk by analysing financial and contract documents as well as specific communications and many more…

**2**

**1.5 CHALLENGES**

**. Data challenges**

The main challenge is information overload, which poses a big problem to access a specific, important piece of information from vast datasets. Semantic and context understanding is essential as well as challenging for summarisation systems due to quality and usability issues. Also, identifying the context of interaction among entities and objects is a crucial task, especially with high dimensional, heterogeneous, complex and poor-quality data.

#### ****. Text related challenges****

Large repositories of textual data are generated from diverse sources such as text steams on the web, communications through mobile and IoT devices. Though ML and NLP have emerged as the most potent and most used technology applied to the analysis of the text and text classification remains the most popular and the most used technique. Text classification could be Multi-Level (MLC) or Multi-Class (MCC). In MCC, every instance could be assigned to only one class label, whereas MLC is a classiﬁcation that assigns multiple labels to a single instance.

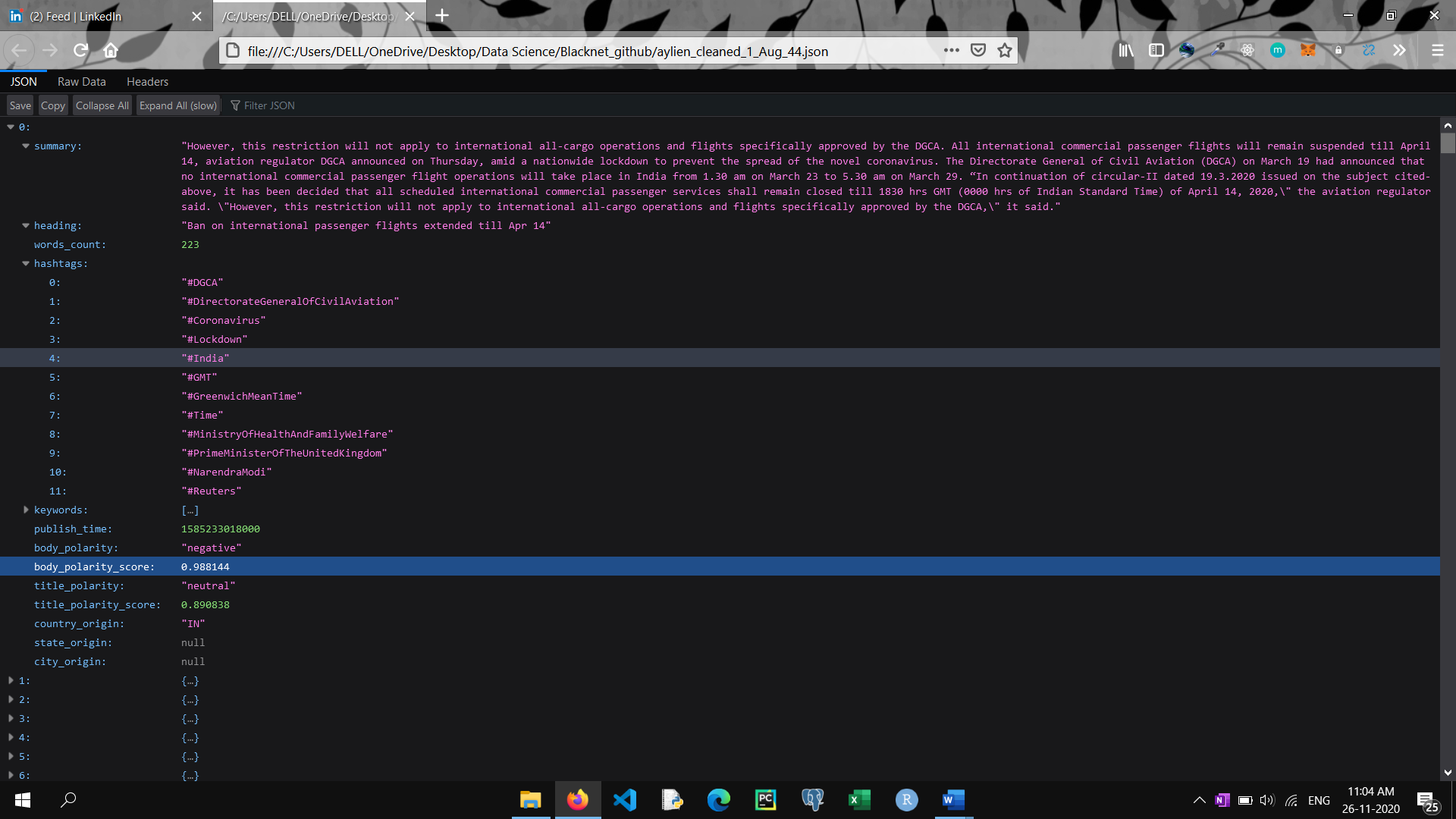
**1.6 DATASET**

Data fetched by the scripts is not ready for use directly. For the idea, roughly 80% of the data was usable. Data contains noise in form of other texts, ads, pictures. The data was converted to the JSON format consisting of about 1,00,000 data information. I used only the essential part of the dataset to provide the insights of the dataset.

The dataset contains the information about the Trending news in year 2020 about the CORONAVIRUS which includes:

* SUMMARY: the summary of the news head lines
* HEADING: the heading of the newspaper
* WORD COUNTS: the number of words in the news
* KEYWORDS: the keywords in the news
* PUBLISH TIME: the time when news was published
* BODY PLOARITY SCORE: positivity of the news body
* TITLE PLOARITY SCORE: positivity of the title
* COUNTRY ORIGIN: where the news is form
* STATE ORIGIN: which state the news is from
* CITY ORIGIN: which city does the news belong

**3**



**1.7 OBJECTIVE**

My main objective for the project was to give the insights of the dataset provided, where I cleaned the data and made it ready to use for them in the readable and extractable format. LDA topic modelling was used in order to make the sense of the words and extract the meaningful topics which were there in the dataset.

I was also asked to find the MOST and LEAST frequent words that were present in the newspapers, articles, and magazines of 2020, so that they can train the Machine Learning model with greater precision to rank the most widely spread and least widely spread news in the city or state on their site and spread awareness of the current and the least spread news (that is not taken care of due to COVID-19) to the people while gaining the traffic for their company.

So, I used some of the technologies like NLP and Gensim to visualise the most and the least popular and related news in the year 2020. I used the headlines, summary, hashtags, polarity and many other parameters to get the words that are related to each other.

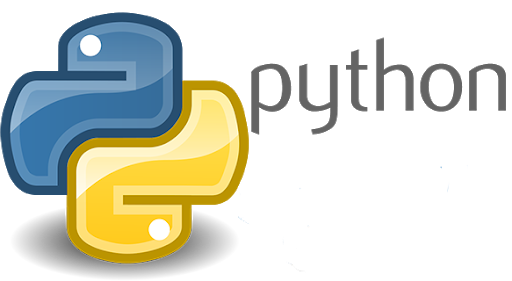
**4**

**CHAPTER**

**TWO**

**TECHNOLOGY**

**2.1 PYTHON**



Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**What PYTHON is used for?**

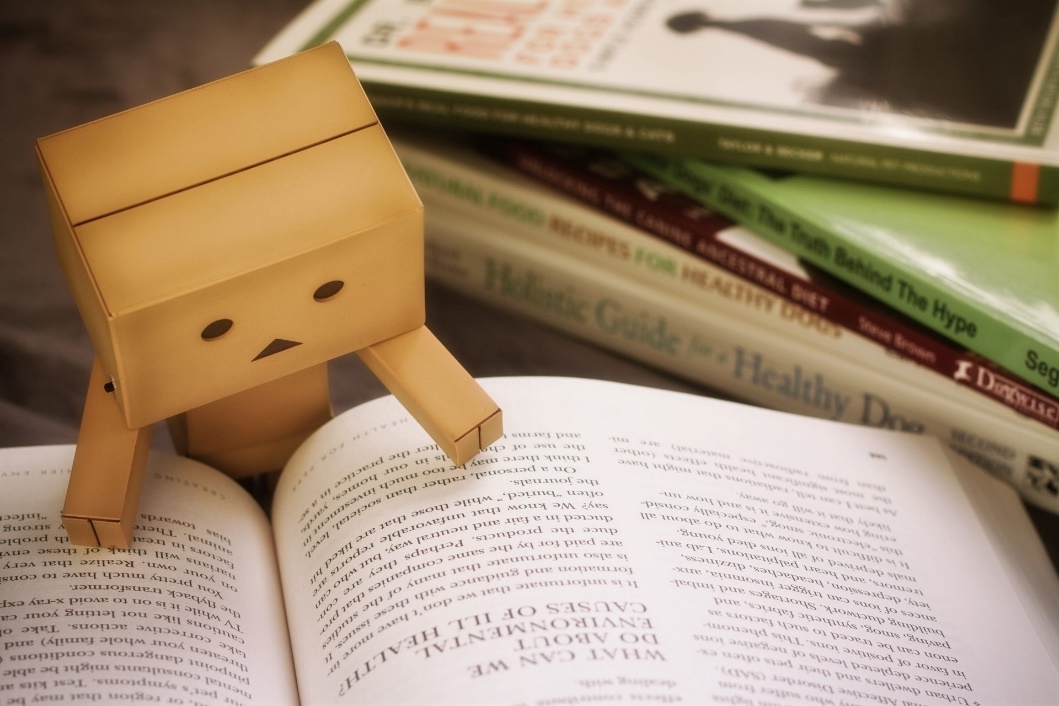
The most basic use case for Python is as a scripting and automation language. Python isn’t just a replacement for shell scripts or batch files; it is also used to automate interaction with web browsers or application GUI or to do system provisioning and configuration in tools such as Ansible and Salt. But scripting and automation represent only the tip of the iceberg with Python.

* General Purpose Programming with PYTHON
* Data Science and Machine Learning with PYTHON
* Web services and RESTful APIs in PYTHON
* Metaprogramming and Code generation with PYTHON

**6**

**2.2 NLP (Natural Language Processing)**

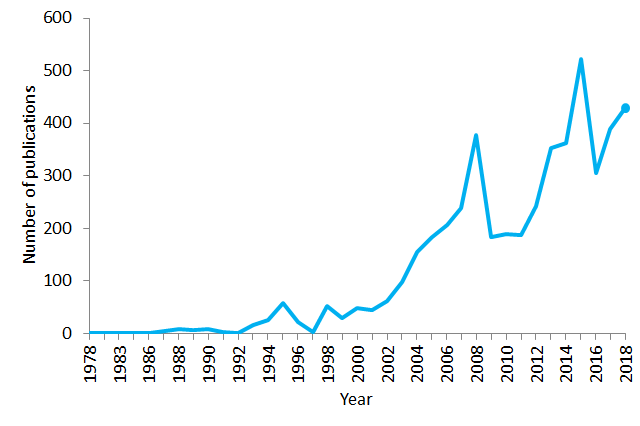
**Natural Language Processing or NLP is a field of Artificial Intelligence that gives the machines the ability to read, understand and derive meaning from human languages.**



Data generated from conversations, declarations or even tweets are examples of unstructured data. **Unstructured data** doesn’t fit neatly into the traditional row and column structure of relational databases, and represent the vast majority of data available in the actual world. It is messy and hard to manipulate. Nevertheless, thanks to the advances in disciplines like machine learning a big revolution is going on regarding this topic. Nowadays it is no longer about trying to interpret a text or speech based on its keywords mechanical way), but about understanding the meaning behind those words (the cognitive way). This way it is possible to detect figures of speech like irony, or even perform sentiment analysis.

NLP is particularly booming in the **healthcare industry**. This technology is improving care delivery, disease diagnosis and bringing costs down while healthcare organizations are going through a growing adoption of electronic health records. The fact that clinical documentation can be improved means that patients can be better understood and benefited through better healthcare. The goal should be to optimize their experience, and several organizations are already working on this.

**7**

Number of publications containing the sentence “natural language processing” in PubMed in the period 1978–2018. As of 2018, PubMed comprised more than 29 million citations for biomedical literature

**Example**:

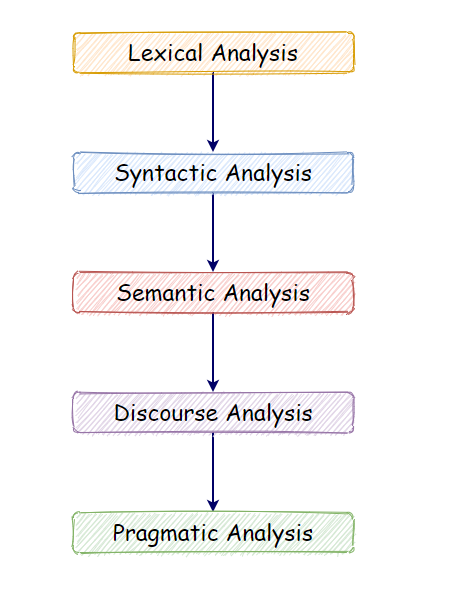
Figure 3: NLP example sentence with the text: “Can you help me with the can?”

In the sentence above, we can see that there are two “can” words, but both of them have different meanings. Here the first “can” word is used for question formation. The second “can” word at the end of the sentence is used to represent a container that holds food or liquid.

Hence, from the examples above, we can see that language processing is not “deterministic” (the same language has the same interpretations), and something suitable to one person might not be suitable to another. Therefore, Natural Language Processing (NLP) has a non-deterministic approach. In other words, Natural Language Processing can be used to create a new intelligent system that can understand how humans understand and interpret language in different situations.

**8**

# **Components of Natural Language Processing (NLP):**



## **. Lexical Analysis:**

With lexical analysis, we divide a whole chunk of text into paragraphs, sentences, and words. It involves identifying and analysing words’ structure.

## **. Syntactic Analysis:**

Syntactic analysis involves the analysis of words in a sentence for grammar and arranging words in a manner that shows the relationship among the words. For instance, the sentence “The shop goes to the house” does not pass.

**9**

## **. Semantic Analysis:**

Semantic analysis draws the exact meaning for the words, and it analyses the text meaningfulness. Sentences such as “hot ice-cream” do not pass.

## **. Disclosure Integration:**

Disclosure integration takes into account the context of the text. It considers the meaning of the sentence before it ends. For example: “He works at Google.” In this sentence, “he” must be referenced in the sentence before it.

## **. Pragmatic Analysis:**

Pragmatic analysis deals with overall communication and interpretation of language. It deals with deriving meaningful use of language in various situations.

**SOME IMPORTANT LIBRARIES:**

## **. NLTK (NATURAL LANGUAGE TOOLKIT):**

The NLTK Python framework is generally used as an education and research tool. It’s not usually used on production applications. However, it can be used to build exciting programs due to its ease of use.

**Features:**

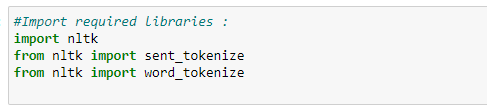
* Tokenization.
* Part of Speech tagging (POS).
* Named Entity Recognition (NER).
* Classification.
* Sentiment analysis.
* Packages of chatbots.

**Use-cases:**

* Recommendation systems.
* Sentiment analysis.
* Building chatbots.
* Text Analysis

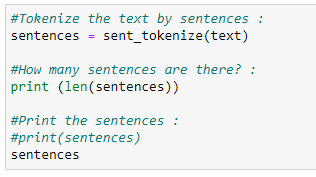
**10**

## **IMPORTING LIBRARIES:**



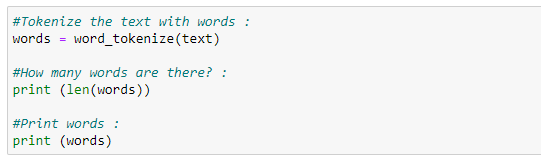
## **SENTENCE TOKENIZING:**

By tokenizing the text, we can get the text as sentences.



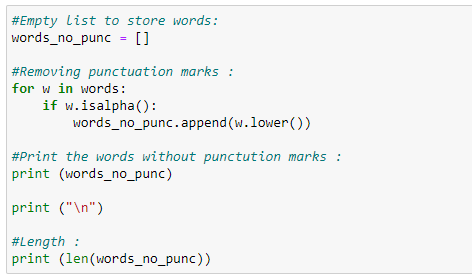
## **WORD TOKENIZING:**

By tokenizing the text, we can get the text as words.

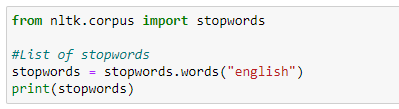


**11**

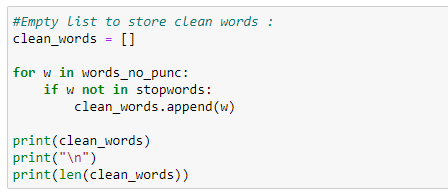
## **REMOVE PUNCTUATION MARKS:**



## **LIST OF STOPWORDS:**



## **REMOVING STOPWORDS:**



**12**

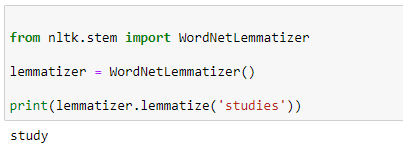
# **STEMMING:**

We use Stemming to normalize words. In English and many other languages, a single word can take multiple forms depending upon context used. For instance, the verb “study” can take many forms like “studies,” “studying,” “studied,” and others, depending on its context. When we tokenize words, an interpreter considers these input words as different words even though their underlying meaning is the same. Moreover, as we know that NLP is about analysing the meaning of content, to resolve this problem, we use stemming.



**LEMMATIZATION:**

Lemmatization tries to achieve a similar base “stem” for a word. However, what makes it different is that it finds the dictionary word instead of truncating the original word. Stemming does not consider the context of the word. That is why it generates results faster, but it is less accurate than lemmatization.

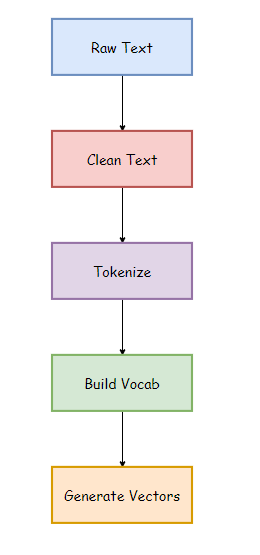


**13**

# **BAG OF WORDS:**

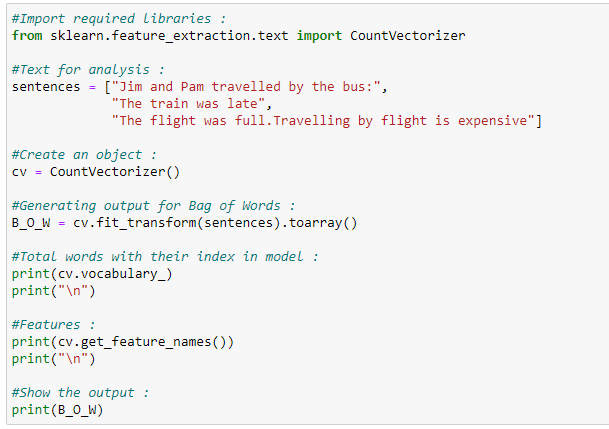


It is a method of extracting essential features from row text so that we can use it for machine learning models. We call it **“Bag”** of words because we discard the order of occurrences of words. A bag of words model converts the raw text into words, and it also counts the frequency for the words in the text. In summary, a bag of words is a collection of words that represent a sentence along with the word count where the order of occurrences is not relevant.



**14**

1. **Raw Text:** This is the original text on which we want to perform analysis.
2. **Clean Text:** Since our raw text contains some unnecessary data like punctuation marks and stopwords, so we need to clean up our text. Clean text is the text after removing such words.
3. **Tokenize:** Tokenization represents the sentence as a group of tokens or words.
4. **Building Vocab:** It contains total words used in the text after removing unnecessary data.
5. **Generate Vocab:** It contains the words along with their frequencies in the sentences.



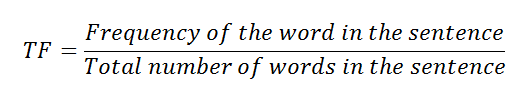
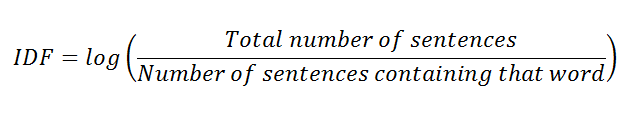
## **LIMITATIONS:**

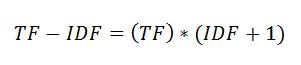
1. **Semantic meaning**: It does not consider the semantic meaning of a word. It ignores the context in which the word is used.
2. **Vector size**: For large documents, the vector size increase, which may result in higher computational time.
3. **Pre-processing**: In pre-processing, we need to perform data cleansing before using it.

**15**

# **TF-IDF**

TF-IDF stands for **Term Frequency — Inverse Document Frequency**, which is a scoring measure generally used in information retrieval (IR) and summarization.





# **WORD EMBEDDINGS WITH GENSIM:**

Word embeddings are a modern approach for representing text in natural language processing.

Word Embedding Algorithms like word2vec and GloVe are key to the state-of-the-art results achieved by neural network models on natural language processing problems like machine translation.

**16**

## **. GENSIM:**

Gensim is an NLP Python framework generally used in topic modeling and similarity detection. It is not a general-purpose NLP library, but it handles tasks assigned to it very well.

**Features:**

* Latent semantic analysis.
* Non-negative matrix factorization.
* TF-IDF.

**Use-cases:**

* Converting documents to vectors.
* Finding text similarity.
* Text summarization.

**2.3 IDE (Integrated Development Environment**)

A code editor is a tool that is used to write and edit code. They are usually lightweight and can be great for learning. However, once your program gets larger, you need to test and debug your code, that's where IDEs come in.

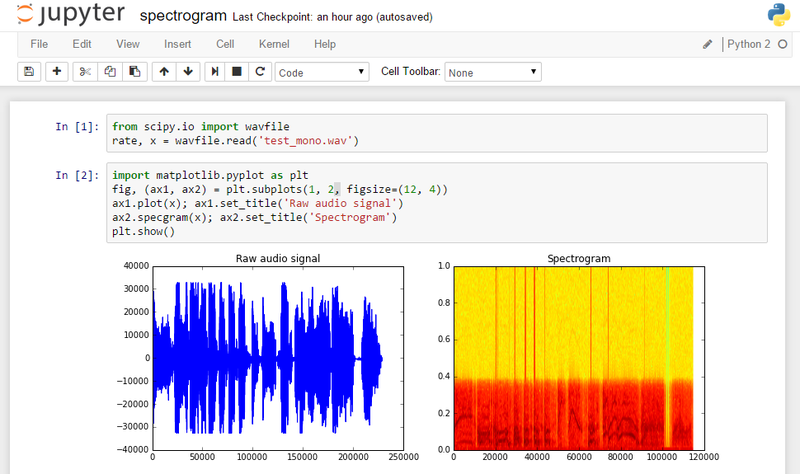
An IDE (Integrated Development Environment) understand your code much better than a text editor. It usually provides features such as build automation, code linting, testing and debugging. This can significantly speed up your work. The downside is that IDEs can be complicated to use.

Because of all the features that IDEs have to offer, they are extremely useful for development: they make your coding more comfortable and this is no different for data science. However, given the fact that there aren’t only the traditional IDEs to consider, but also new tools, such as notebooks, you might be wondering which development environment to use when you’re just starting out with data science.

**17**

## **JUPYTER NOTEBOOK:**

**Features** the Jupyter Notebook supports markdowns, allowing you to add HTML components from images to videos. Thanks to Jupyter, you can easily see and edit your code in order to create compelling presentations. For instance, you can use data visualization libraries like Matplotlib and Seaborn and show your graphs in the same document where your code is. Besides all of this, you can export your final work to PDF and HTML files, or you can just export it as a .py file. In addition, you can also create blogs and presentations from your notebooks.



**18**

**DATA CLEANING**

**and**

**ANALYSING**

**3.1 CLEANING THE DATA**

Incorrect or inconsistent data leads to false conclusions. And so, how well you clean and understand the data has a high impact on the quality of the results.



Example:

*In the business world, incorrect data can be costly. Many companies use customer information databases that record data like contact information, addresses, and preferences. For instance, if the addresses are inconsistent, the company will suffer the cost of resending mail or even losing customers.*

* **Some of the tasks you should look for CLEANING DATA:**
* Make sure numbers are stored as numerical data types. A date should be stored as a date object, or a Unix timestamp (number of seconds), and so on.
* **Remove white spaces:** Extra white spaces at the beginning or the end of a string should be removed.
* Our duty is to not only recognize the typos but also put each value in the same standardized format.
* Scaling means to transform your data so that it fits within a specific scale, such as 0–100 or 0–1.
* Given the fact the missing values are unavoidable leaves us with the question of what to do when we encounter them. Ignoring the missing data is the same as digging holes in a boat; It will sink.
* They are values that are significantly different from all other observations. Any data value that [lies more than (1.5 \* IQR) away from the Q1 and Q3 quartiles](https://medium.com/omarelgabrys-blog/statistics-probability-exploratory-data-analysis-714f361b43d1#48f7) is considered an outlier.

**20**

The data provide to me was in json format which is not directly used in the IDE we have to first convert the data into a DATAFRAME and then analyse the data.

Keywords, Hashtags, Headings and Summary were the important features which with we can find the correlation between the words. But the Keywords and Hashtags were in the LIST format and Summary and Headings were in the text, so I need to convert all the features into the same data type so that they can be used on the same platform.

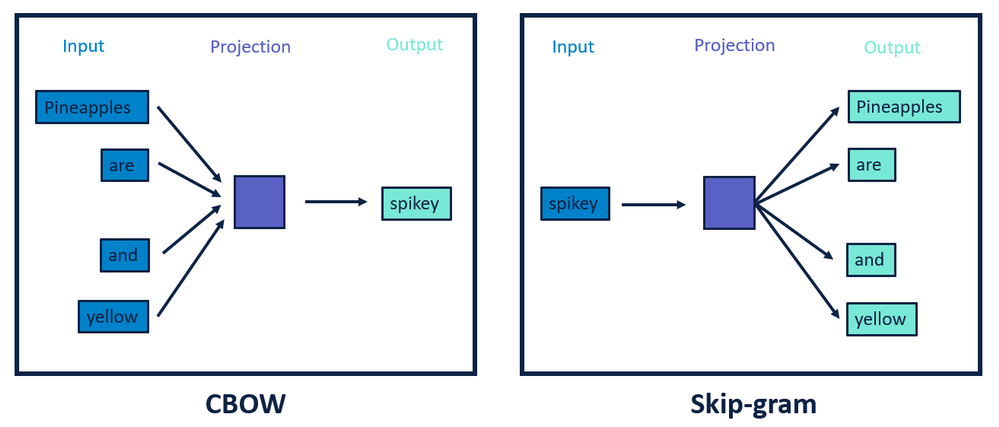
Then checked the rows that have NULL values as those values can result in false prediction of the model. There were 2 features (State Origin and City Origin) that have 380 NULL values and are to be removed, as the data set is large, we don’t have to worry about the minimisation of data or computing any other value in them.

Then the Keywords and Hashtags features were UNLISTED and then combined into the data frame. Afterwards the Summary and Headings were CONCATINATED and were assigned to one column.

Other features were Dropped as they were Numerical and won’t count for the FEATURE SELECTION. Then all the features (Keywords, Hashtags, Summary and Headings were combined together in a single column and NLP is then applied to them to find the meaningful values of data. STOPWORDS were removed as there were some of the STOPWORDS that were not present in the STOPWORDS list we added them manually like ['per', 'due', 'daily', 'one', 'also', 'day', 'time', 'new', 'r', 'home', 'year', 'month', 'social']. Then all the numbers are removed and the txt data is converted into the LOWER form and LEMMATIZED.

**21**

**3.2 WORD2VEC**



**Word Embedding:**

Word embedding is nothing fancy but **methods to represent words in a numerical way**. More specifically, methods to map vocabularies to vectors. The most straightforward method could be using one-hot encoding to map each word to a one-hot vector.

**Word2Vec** embedding approach, developed by **Tomas Mikolov**, is considered the state of the art. Word2Vec approach uses deep learning and neural networks-based techniques to convert words into corresponding vectors in such a way that the semantically similar vectors are close to each other in N-dimensional space, where N refers to the dimensions of the vector.

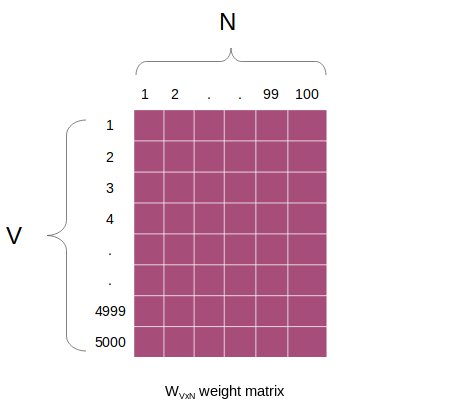
Word2Vec model comes in two flavours: Skip Gram Model and Continuous Bag of Words Model (CBOW).

In the **Skip Gram** model, the context words are predicted using the base word. For instance, given a sentence "I love to dance in the rain", the skip gram model will predict "love" and "dance" given the word "to" as input.

On the contrary, the **CBOW** model will predict "to", if the context words "love" and "dance" are fed as input to the model. The model learns these relationships using deep neural networks.

The model I used for is GENSIM with WORD2VEC as it is extremely straightforward to create Word2Vec model. The word list is passed to the Word2Vec class of the gensim.models package. We need to specify the value for the min\_count parameter. A value of 2 for min\_count specifies to include only those words in the Word2Vec model that appear at least twice in the corpus.

**22**



We know that the Word2Vec model converts words to their corresponding vectors. The vector contains the vector representation for the word "coronavirus". By default, a hundred-dimensional vector is created by Gensim Word2Vec. This is a much, much smaller vector as compared to what would have been produced by bag of words. If we use the bag of words approach for embedding the article, the length of the vector for each will be 1206 since there are 1206 unique words with a minimum frequency of 150. If the minimum frequency of occurrence is set to 1, the size of the bag of words vector will further increase. On the other hand, vectors generated through Word2Vec are not affected by the size of the vocabulary.

The contextual information of the words is not lost using Word2Vec approach. We can verify this by finding all the words similar to the word "covid".

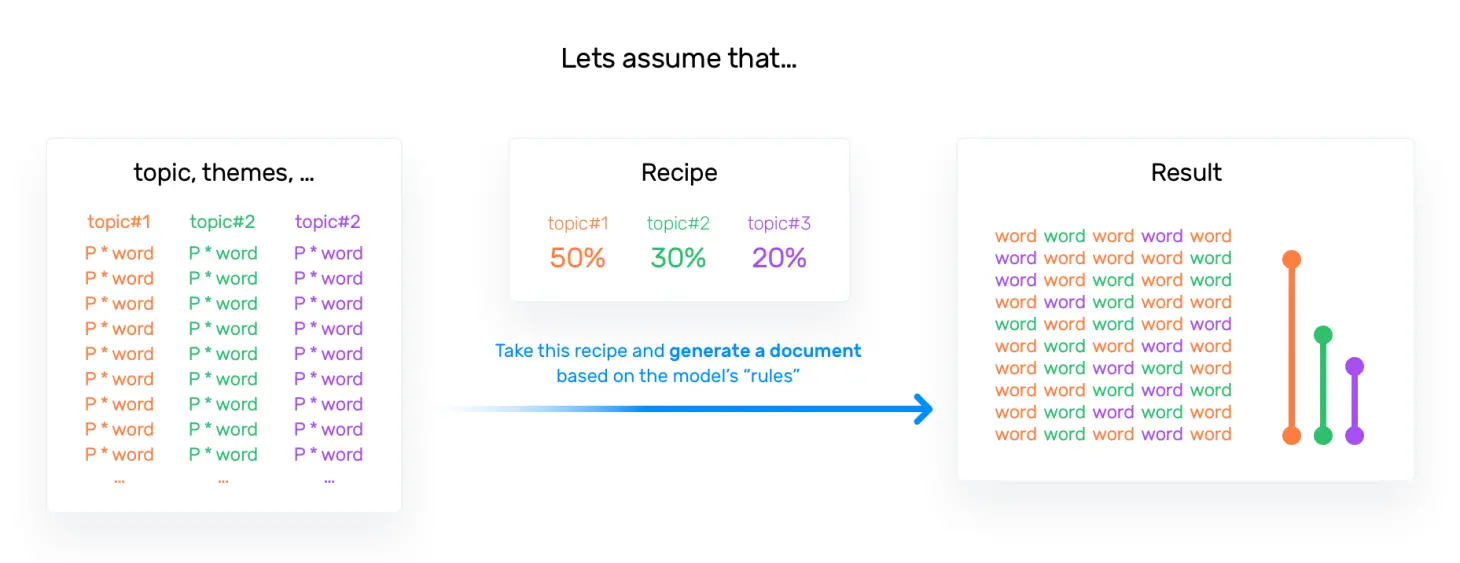
We can find the vectors too that are there in the word “coronavirus” and “covid” by using “model”. Then we saw the vocabulary of the trained model and then converted that into a data frame so that we can find the correlation between the words.

**23**

# **3.3 Latent Dirichlet Allocation (LDA)**

Topic modelling is a machine learning technique that automatically analyses text data to determine cluster words for a set of documents. This is known as ‘unsupervised’ machine learning because it doesn’t require a predefined list of tags or training data that’s been previously classified by humans.

*Latent Dirichlet Allocation (LDA)* is based on the underlying assumptions: the distributional hypothesis, (i.e. similar topics make use of similar words) and the statistical mixture hypothesis (i.e. documents talk about several topics) for which a statistical distribution can be determined. The purpose of LDA is mapping each document in our corpus to a set of topics which covers a good deal of the words in the document.

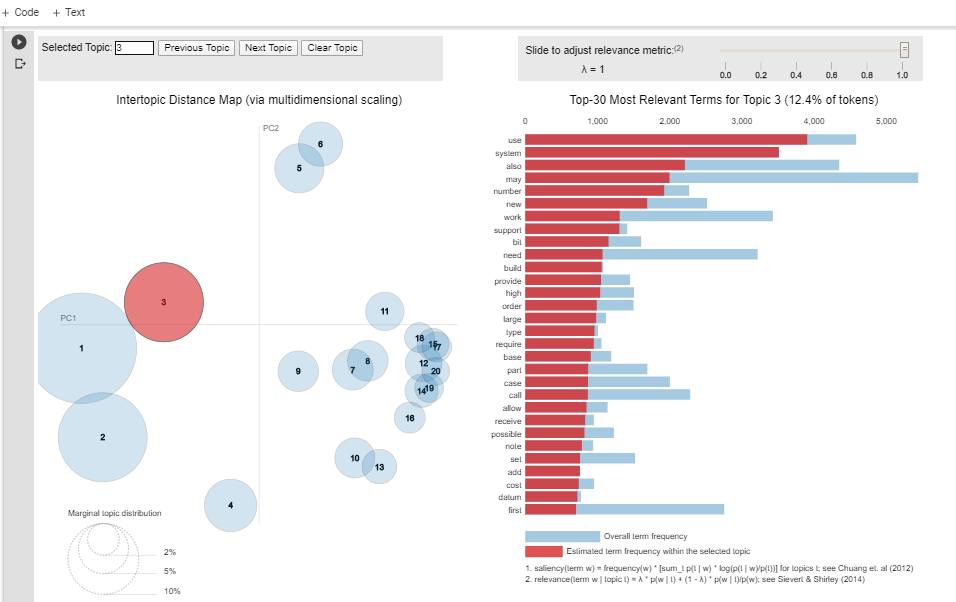


What LDA does in order to map the documents to a list of topics is assign topics to arrangements of words, e.g. n-grams such as *best player* for a topic related to sports. This stems from the assumption that documents are written with arrangements of words and that those arrangements determine topics. LDA also ignores syntactic information and treats documents as bags of words. It also assumes that all words in the document can be assigned a probability of belonging to a topic. That said, the goal of LDA is to determine the mixture of topics that a document contains.

The next step is to examine the produced topics and the associated keywords. There is no better tool than pyLDAvis package’s interactive chart and is designed to work well with jupyter notebooks.

**24**

**3.4 VISUAL ANALYSIS**



Each bubble on the left-hand side represents topic. The larger the bubble, the more prevalent or dominant the topic is. Good topic model will be fairly big topics scattered in different quadrants rather than being clustered on one quadrant.

* The model with too many topics will have many overlaps, small sized bubbles clustered in one region of chart.
* If you move the cursor the different bubbles you can see different keywords associated with topics.

**How to find optimum number of topics?**

* One approach to find optimum number of topics is build many LDA models with different values of number of topics and pick the one that gives highest coherence value.
* If you see the same keywords being repeated in multiple topics, it’s probably a sign that the ‘k’ is too large.
* Sometimes topic keyword may not be enough to make sense of what topic is about. So, for better understanding of topics, you can find the documents a given topic has contributed the most to and infer the topic by reading the documents.
* Finally, one needs to understand the volume and distribution of topics in order to judge how widely it was discussed.

**25**

**RESULT**

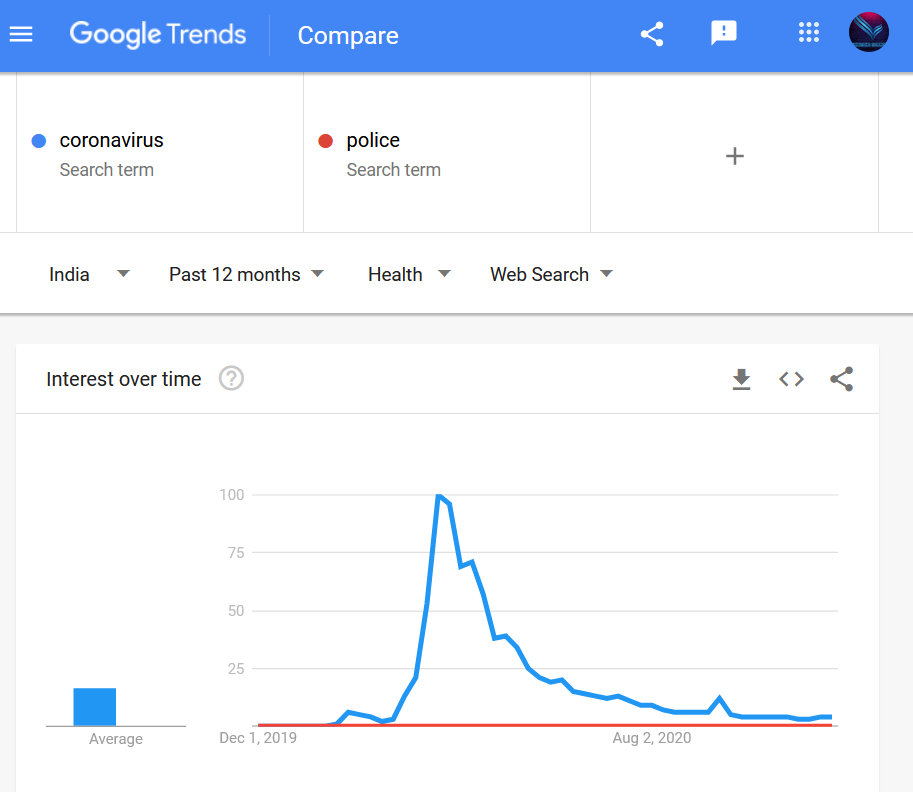
**&**

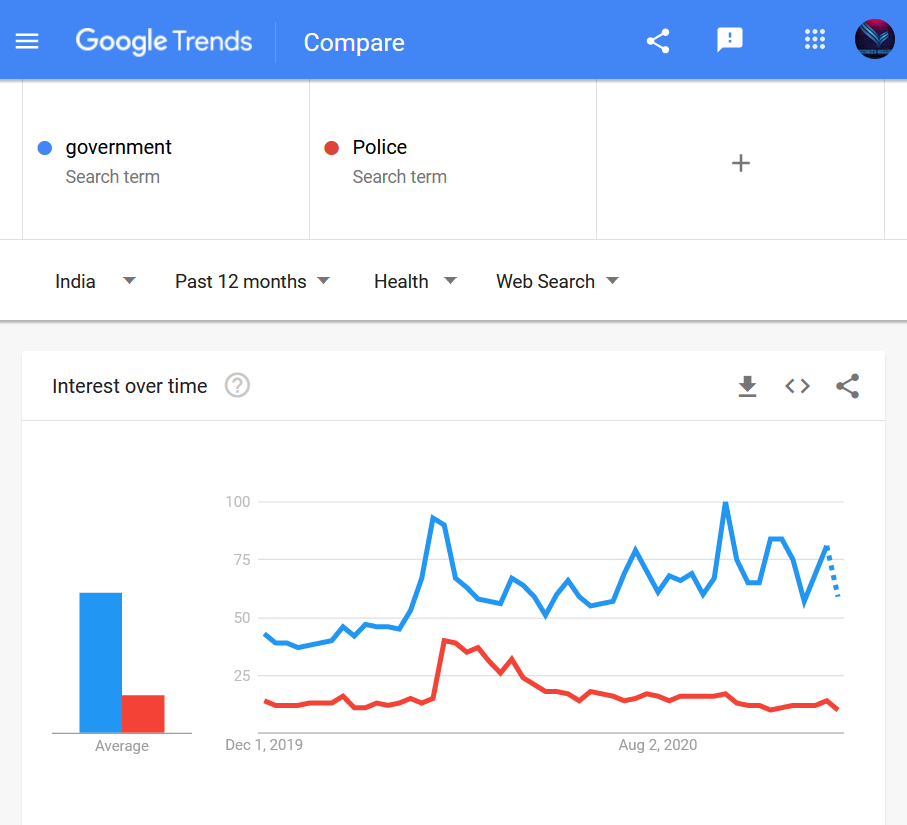
**OBSERVATION**

**4.RESULT & OBSERVATION**

As Unlock-2 begins from July 1, India’s COVID-19 tally climbed to 5,66,840 with nearly 66% cases reported in June alone, and a concerned Prime Minister Narendra Modi said it is a cause of worry that people are not strictly adhering to rules and precautions as they did during the lockdown. From June 1, when relaxations were introduced under Unlock-1, the country has reported 3,76,305 cases. Maharashtra, Tamil Nadu, Delhi and Gujarat account for nearly two-third of the total cases till now.

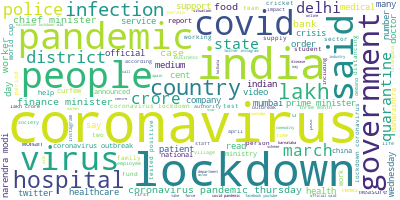
The data contains the news headline, hashtags, summary and keywords from NOVEMBER 2019-JUNE 2020. So, I have tried to capture the most frequently and least frequent words from combining these fields. I got to know that “CORONAVIRUS”,” LOCKDOWN”, “COVID”,” INDIA”,” PANDEMIC”,” GOVERNMENT” are some of the most frequently used words. Whereas “POLICE”,” POOR”,” BANK”,” OFFICIAL” are some the least used words according to the data.





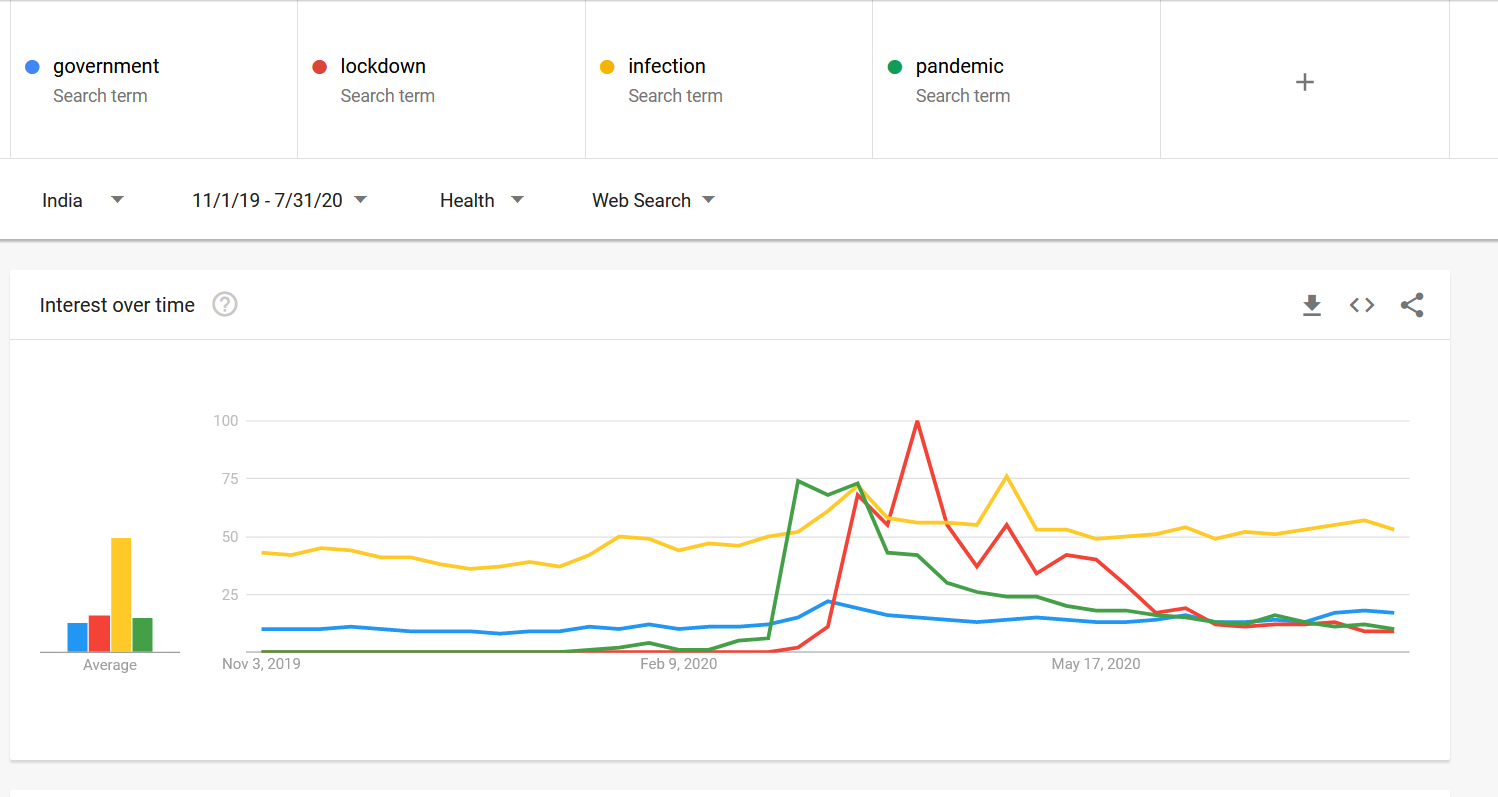
**27**

As we can see the GOOGLE TRENDS between some of the words like “GOVERNMENT – POLICE” there is much more trends of CORONAVIRUS and similarly between the “CORONAVIRUS – POLICE” there is negligible trend of POLICE.



This is the WORDCLOUD that shows some of the trending and most popular words that were the part of the dataset. As we can see that “CORONAVIRUS”,” GOVERNMENT”,” LOCKDOWN”,” INFECTION”,” PANDEMIC” and many others are the words that carry more weightage and were mostly used in the 8 months (NOVEMBER 2019 -JUNE 2020).

The TOPIC COHERENCE must lie between **0.3-0.6** and first the base model coherence was **0.3429** when topics selected were 8, then after tuning the model and coming up with the 20 topics the score increases to **0.4402** which is great and showing the relevance between the topics. This was an increase of **28%**.

This shows that the result we are getting while doing the analysis of the dataset is CORRECT! 

**28**

**CONCLUSIONS**

**&**

**FUTURE SCOPE**

**5.Conclusions & Future Scope**

I tried my best to analysis the data with great accuracy. With the help of NLTK, GENSIM, WORDCLOUD and pyLDAvis I was able to get the clear insights of the dataset and was able to put my skills in the right direction.

This report concludes the concepts of NLP and DATA SCIENCE by implementation of the same in the project. This shows us that NLP can be used in many sectors like Business, Health, Government and many other to get the clear insights of the human data that to in human form. The insights collected from the data will be used by the company to find the most popular and least popular spread of news and then rank the news in their website according to the analysis so as to gain more traffic and also put the least interested news into the light so that those news will also get the eyes of media and people that were not taken care of due to COVID\_19.With the topic coherence of **0.44** I was able to maintain the descent range of topic with **20** different.

**Natural Language Processing** or NLP is a field of Artificial Intelligence that gives the machines the ability to read, understand and derive meaning from human languages.

* This project can be extended to many fields as NLP is booming in the HEALTH SECTOR as NLP enables the recognition and **prediction of diseases** based on electronic health records and patient’s own speech. This capability is being explored in health conditions that go from cardiovascular diseases to depression and even schizophrenia. For example, Amazon Comprehend Medical is a service that uses NLP to extract disease condition medications and treatment outcomes from patient notes, clinical trial reports and other electronic health records.
* Organizations can determine what customers are saying about a service or product by identifying and extracting information in sources like social media, SENTIMENTAL ANALYSIS can provide a lot of information about customers choices and their decision drivers.
* Companies like Yahoo and Google filter and classify your emails with NLP by analysing text in emails that flow through their servers and **stopping spam** before they even enter your inbox.
* Amazon’s Alexa and Apple’s Siri are examples of intelligent **voice driven interfaces** that use NLP to respond to vocal prompts and do everything like find a particular shop, tell us the weather forecast, suggest the best route to the office or turn on the lights at home.

**30**

**6.REFERNECES**

**RESEARCH PAPERS & LINKS**

# https://thinkinfi.com/guide-to-build-best-lda-model-using-gensim-python/

### Maria Barrett, Joachim Bingel, Nora Hollenstein, Marek Rei, Anders Søgaard, Sequence Classification with Human Attention ,2018

# <https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0>

# Linguistically-Informed Self-Attention for Semantic RoleLabeling [Emma Strubell](https://arxiv.org/search/cs?searchtype=author&query=Strubell%2C+E), [Patrick Verga](https://arxiv.org/search/cs?searchtype=author&query=Verga%2C+P), [Daniel Andor](https://arxiv.org/search/cs?searchtype=author&query=Andor%2C+D), [David Weiss](https://arxiv.org/search/cs?searchtype=author&query=Weiss%2C+D), [Andrew McCallum](https://arxiv.org/search/cs?searchtype=author&query=McCallum%2C+A), Submitted on 23 Apr 2018 ([v1](https://arxiv.org/abs/1804.08199v1)), last revised 12 Nov 2018

### [A Hierarchical Multi-task Approach for Learning Embeddings from Semantic Tasks](https://arxiv.org/abs/1811.06031), by Victor Sanh, Thomas Wolf, and Sebastian Ruder, Submitted on 14 Nov 2018 ([v1](https://arxiv.org/abs/1811.06031v1)), last revised 26 Nov 2018

# <https://medium.com/swlh/topic-modeling-lda-mallet-implementation-in-python-part-2-602ffb38d396>

# <https://medium.com/@kurtsenol21/topic-modeling-lda-mallet-implementation-in-python-part-1-c493a5297ad2>

1. Distributed Representations of Words and Phrasesand their Compositionality, Tomas Mikolov, Submitted on 16 Jan 2013 ([v1](https://arxiv.org/abs/1301.3781v1)), last revised 7 Sep 2013

**[9]** <https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-> modeling-in-python/

**31**

**BOOKS**

1. Data science programming All-In-One for Dummies,John Paul Mueller, Luca Massaron, GDE
2. Data Science Strategy for Dummies, Ulrika Jagare
3. Data Warehousing for Dummies, Thomas C.Hammergern, Alan R. Simon
4. Statistics for Big Data for Dummies, Alan Anderson,PhD
5. The GDPR & Managing Data Risk, Andrew Moore
6. Machine Learning for Dummies, Judith Hurwitz, Daniel Kirsch

## Discover The Fastest Growing Platform For Professional Machine learningWith Step-By-Step Tutorials and End-To-End Projects, Jason Browniee

# Natural Language Processing with Python,by [Steven Bird](http://www.stevenbird.net/), [Ewan Klein](http://homepages.inf.ed.ac.uk/ewan/) and [Edward Loper](http://ed.loper.org/).

# Foundations of Statistical Natural Language Processing,by[Christopher Manning](https://nlp.stanford.edu/manning/) and [Hinrich Schütze](http://www.cis.uni-muenchen.de/schuetze/).

# Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition,by [Dan Jurafsky](https://www.goodreads.com/author/show/5802762.Dan_Jurafsky) and [James H. Martin](https://www.goodreads.com/author/show/464819.James_H_Martin)

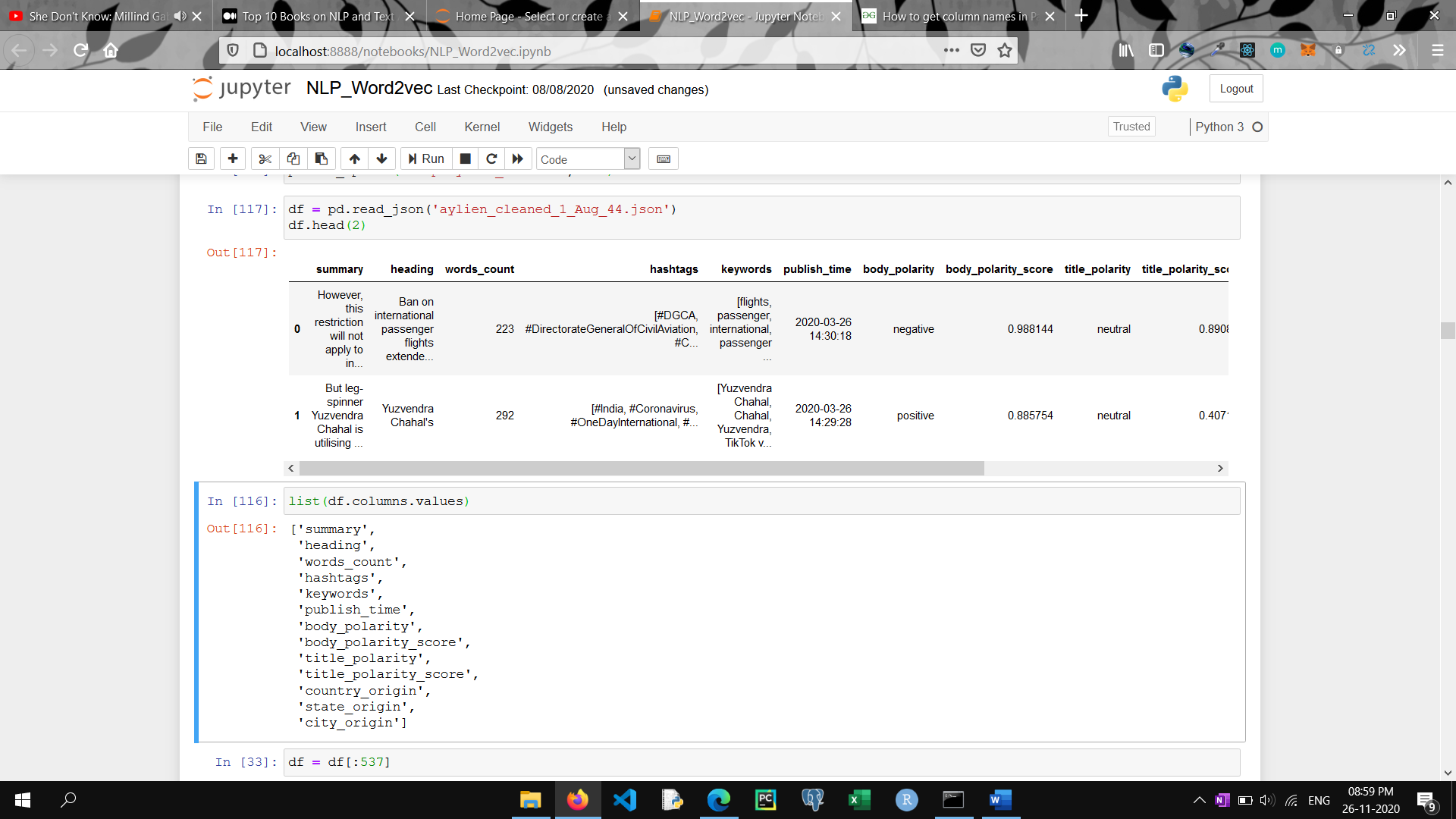
**32**

**APPENDIX**

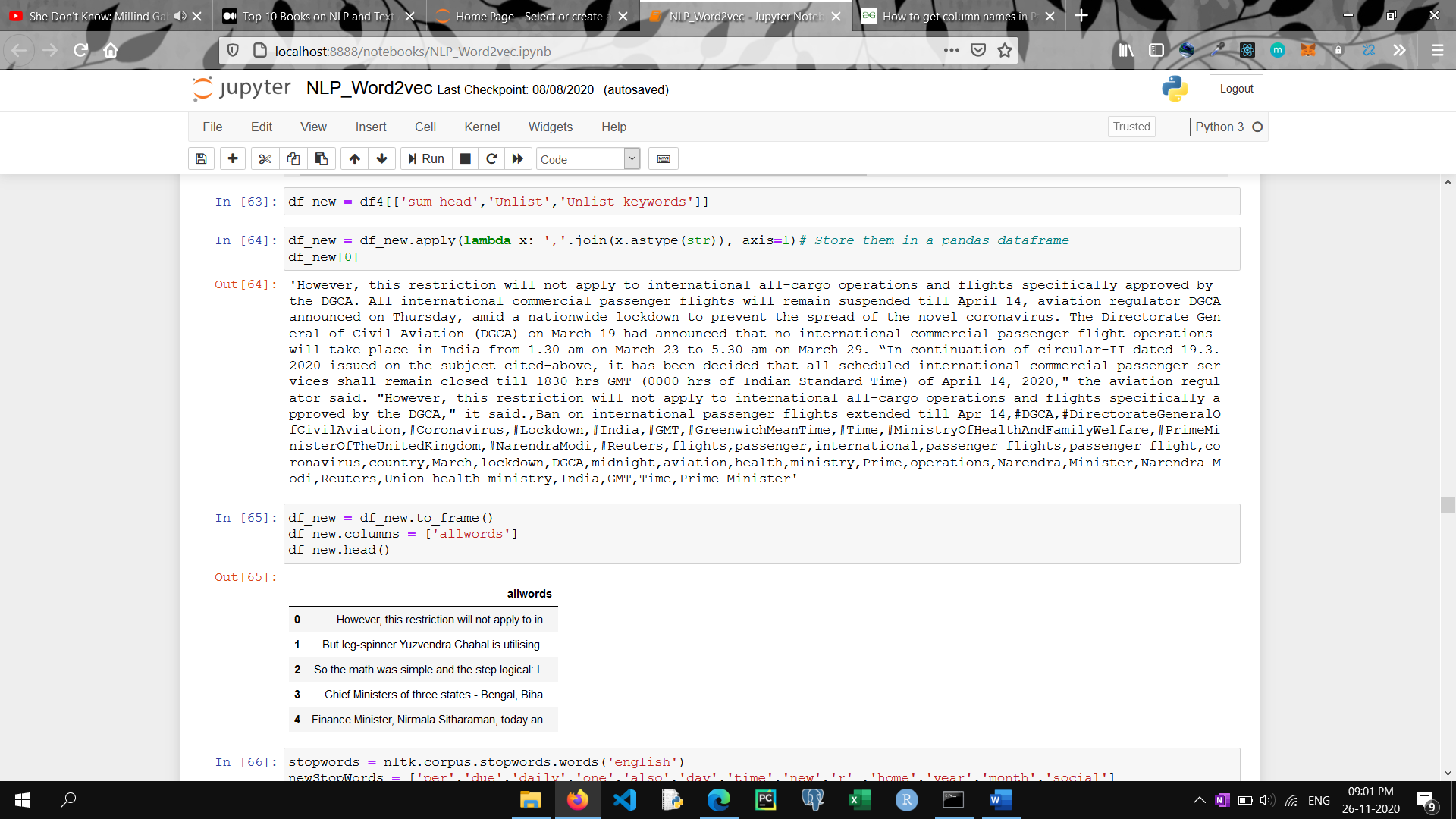
**SCREENSHORT**

**A. SCREENSHORT**

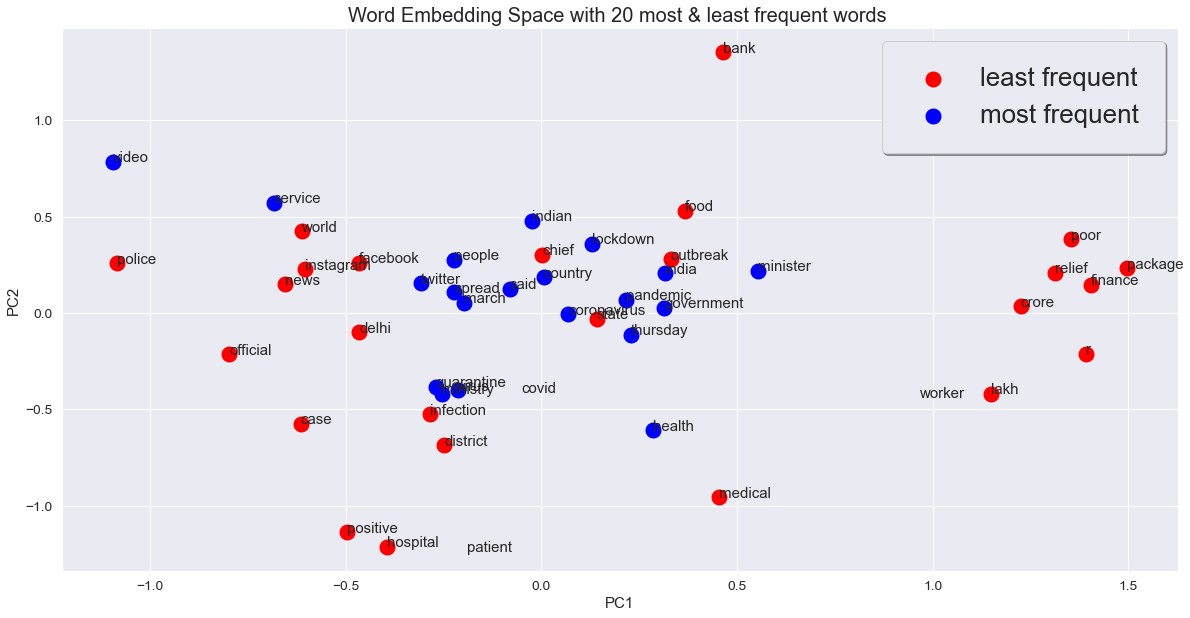
**DATASET:**



**DATA CLEANING:**

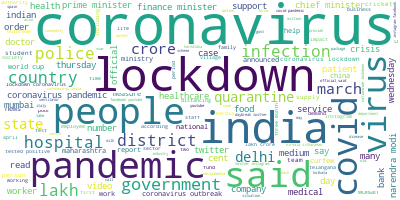


**34**

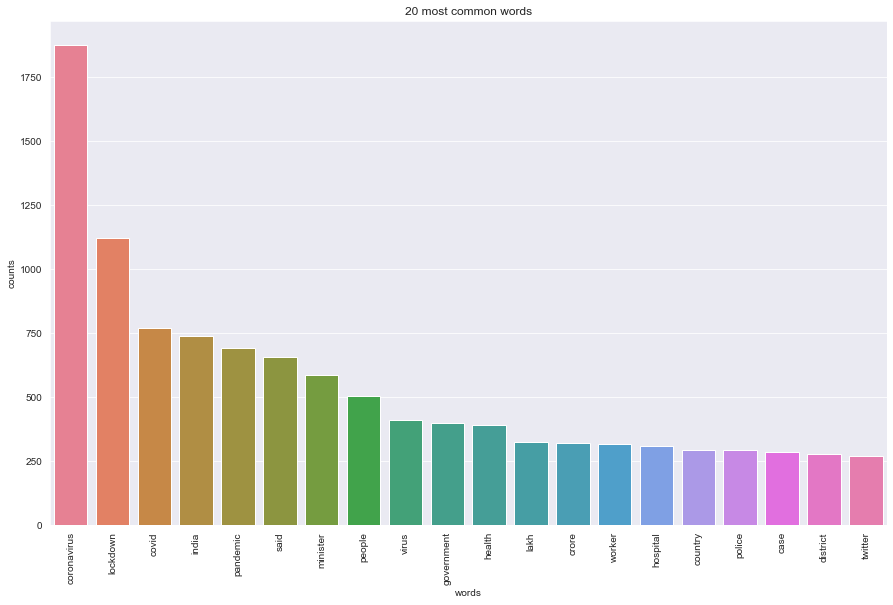
****

**35**

**MOST FREQUENT WORDS IN WORDCLOUD**

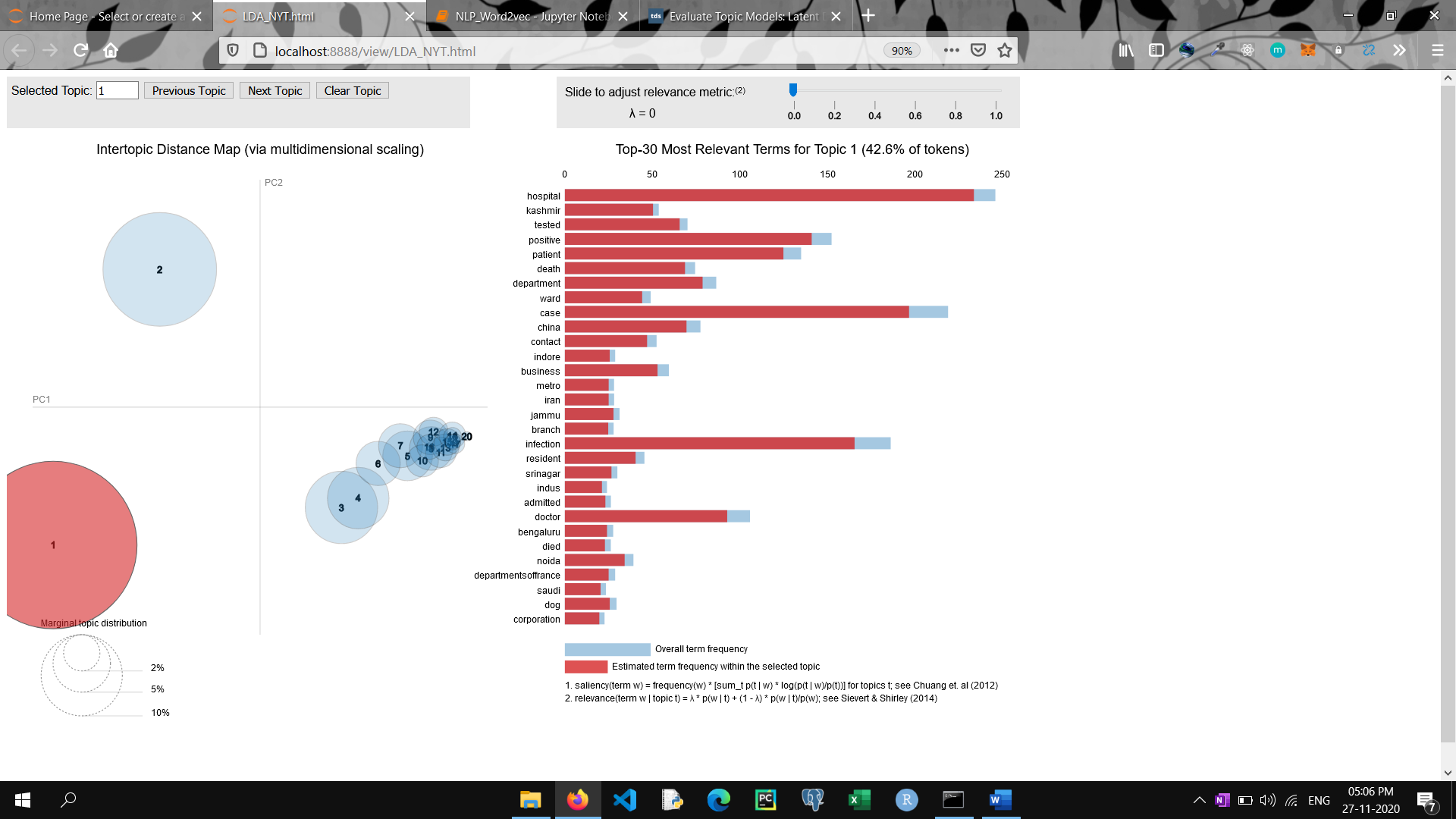
****

**TOP 20 MOST FREQUENT WORDS IN BARGRAPH:**

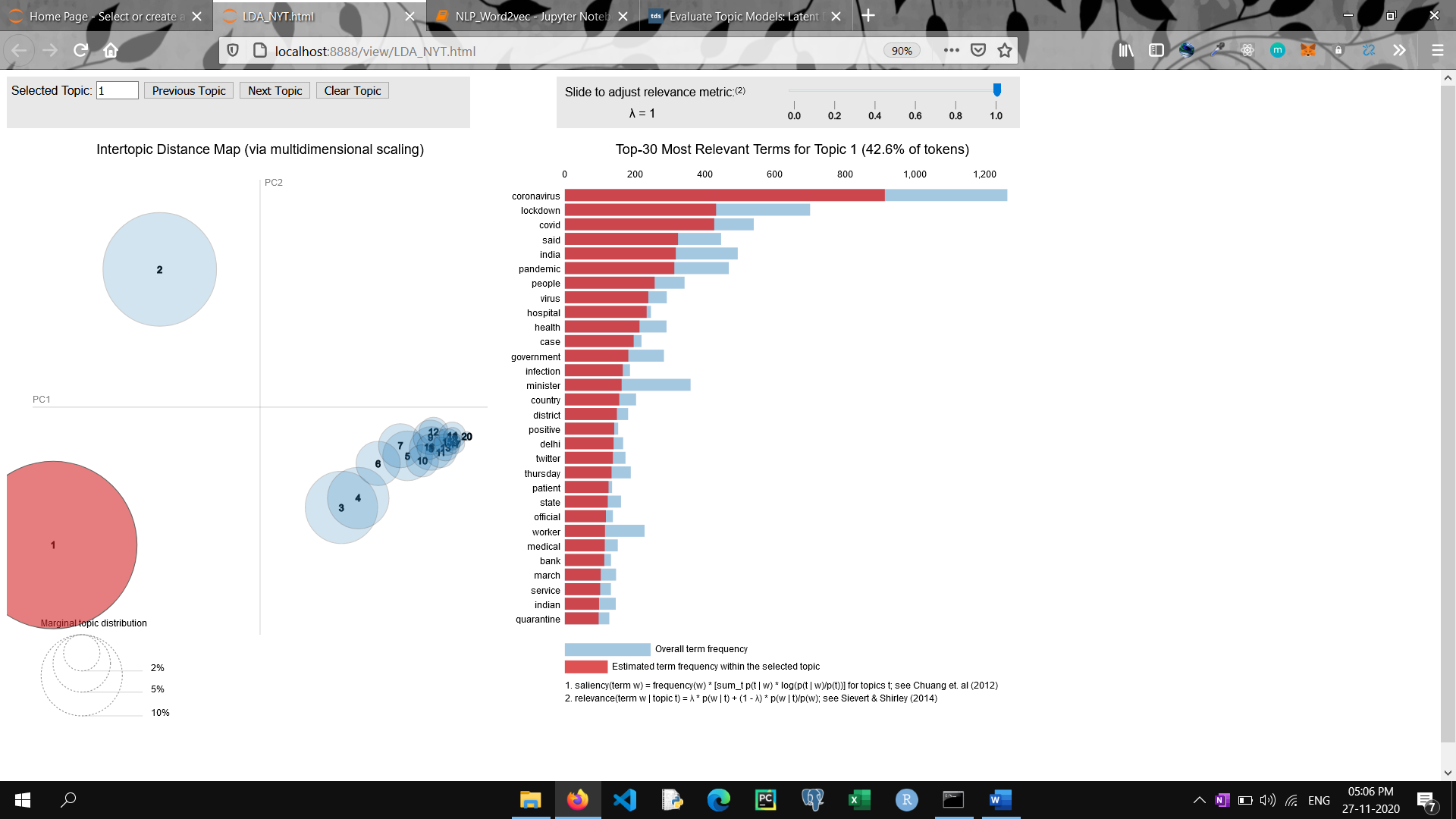
****

**36**

**LEAST RELEVANCE WITH TOPIC 1**



**MOST RELEVANCE WITH TOPIC 1**



**37**

**APPENDIX**

**SOURCE CODE**

**B. SOURCE CODE**

***# # NATURAL LANGUAGE PROCESSING   
#   
# - This is a project that contains the news of the year 2020 with heading , summary , keywords , hashtags , publish\_time, body\_polarity , body\_polarity\_score , title\_polarity , title\_polarity\_score , country\_origin , state\_origin , city\_origin.  
#   
# - The main aim of the project is to analysis the whole dataset and find the most frequent and least frequent words in them   
#   
# - I have used Word2vec for converting the words into vectors and joining the columns [heading , summary , hashtags and keywords] to cleaned them and converted them to the list of lists.  
#   
# - Also used LDA for model making and have used pyLDA for visualisation.  
  
# # Main Project :*****import** pandas **as** pd  
**from** gensim.models **import** Word2Vec   
**import** numpy **as** np  
**import** warnings  
warnings.filterwarnings(**"ignore"**)  
warnings.warn(**"deprecated"**, DeprecationWarning)  
**import** nltk  
**import** re  
nltk.download(**'stopwords'**)  
  
  
*# pd.set\_option('display.max\_rows',None)*pd.set\_option(**'display.max\_columns'**,**None**)  
  
  
df = pd.read\_json(**'aylien\_cleaned\_1\_Aug\_44.json'**)  
df.head(2)  
  
list(df.columns.values)  
  *39*

df = df[:537]

df2 = df.copy()  
  
df1 = df.hashtags.copy()  
  
df1.shape  
df.isnull().sum()  
  
df1[1]

**for** i **in** range(len(df1)):  
 df1[i] = **','**.join(map(str, df1[i]))  
  
df1.head()  
df3 = df1.to\_frame()  
df3.columns = [**'Unlist'**]  
df3.head()

df4 = pd.concat([df3,df2],axis=1)  
  
df4.head()  
df2 = df2.keywords.copy()  
  
df2.head()  
df2[1]  
**for** i **in** range(len(df1)):  
 df2[i] = **','**.join(map(str, df2[i]))  
  
df2.head()  
  
df3 = df2.to\_frame()  
 *40*df3.head()  
df3.columns = [**'Unlist\_keywords'**]  
  
df4 = pd.concat([df3,df4],axis=1)  
  
df4.head()  
df5 = df4[[**'summary'**,**'heading'**]]  
df5 = df5.apply(**lambda** x: **','**.join(x.astype(str)), axis=1)*# Store them in a pandas dataframe*df5[0]

sum\_head = pd.DataFrame({**'sum\_head'**: df5})  
sum\_head.head()df4 = pd.concat([sum\_head,df4],axis=1)  
df4.head()  
  
df4.drop([**'keywords'**,**'hashtags'**,**'summary'**,**'heading'**],axis=1,inplace=**True**)  
df4.head()  
df\_new = df4[[**'sum\_head'**,**'Unlist'**,**'Unlist\_keywords'**]]  
  
df\_new = df\_new.apply(**lambda** x: **','**.join(x.astype(str)), axis=1)*# Store them in a pandas dataframe*df\_new[0]  
  
df\_new = df\_new.to\_frame()  
df\_new.columns = [**'allwords'**]  
 *41*

df\_new.head()

stopwords = nltk.corpus.stopwords.words(**'english'**)  
newStopWords = [**'per'**,**'due'**,**'daily'**,**'one'**,**'also'**,**'day'**,**'time'**,**'new'**,**'r'** ,**'home'**,**'year'**,**'month'**,**'social'**]  
stopwords.extend(newStopWords)  
stop\_words = set(stopwords)  
  
**import** re  
**import** nltk  
nltk.download(**'stopwords'**)  
  
**from** nltk.corpus **import** stopwords  
**from** nltk.stem **import** WordNetLemmatizer   
lemmatizer = WordNetLemmatizer()   
corpus = []  
  
newStopWords = [**'per'**,**'due'**,**'daily'**,**'one'**,**'also'**,**'day'**,**'time'**,**'new'**,**'r'** ,**'home'**,**'year'**,**'month'**,**'social'**]  
  
**for** i **in** range(0, len(df\_new)):  
 review = re.sub(**'[^a-zA-Z]'**, **' '**, df\_new[**'allwords'**][i])  
 review = review.lower()  
 review = review.split()  
   
 review = [lemmatizer.lemmatize(word) **for** word **in** review **if not** word **in** stop\_words]  
 review = **' '**.join(review)  
 corpus.append(review)  
  
corpus[0]  
df\_clean = pd.DataFrame(corpus,columns = [**'text'**])  
df\_clean.head()  
  
sent = [row.split(**' '**) **for** row **in** df\_clean[**'text'**]]  
**from** gensim.models **import** Word2Vec   
model = Word2Vec(sent, min\_count=150,size= 100,workers=3, window =20, sg = 1)  
 *42*model.most\_similar(**'coronavirus'**)[:5]  
model.wv.index2entity[:20] *## top 20 most frequent words*model[**'coronavirus'**]  
words=list(model.wv.vocab)  
print(words)  
X=model[model.wv.vocab]  
df=pd.DataFrame(X)  
df.shape  
df.head()  
 *#Computing the correlation matrix*X\_corr=df.corr()  
  
*#Computing eigen values and eigen vectors*values,vectors=np.linalg.eig(X\_corr)  
  
*#Sorting the eigen vectors coresponding to eigen values in descending order*args = (-values).argsort()  
values = vectors[args]  
vectors = vectors[:, args]  
  
*#Taking first 2 components which explain maximum variance for projecting*new\_vectors=vectors[:,:2]

*#Projecting it onto new dimesion with 2 axis*neww\_X=np.dot(X,new\_vectors)  
  
len(model.wv.index2entity[:])  
  
**import** seaborn **as** sns  
sns.set\_style(**'darkgrid'**)  
  
**import** matplotlib.pyplot **as** plt  
plt.figure(figsize=(20,10))

*43*

plt.scatter(neww\_X[23:,0],neww\_X[23:,1],linewidths=10,color=**'red'**,label=**'least frequent'**)

plt.scatter(neww\_X[:20,0],neww\_X[:20,1],linewidths=10,color=**'blue'**,label = **'most frequent'**)  
plt.xlabel(**"PC1"**,size=15)  
plt.ylabel(**"PC2"**,size=15)  
plt.legend(fancybox=**True**, framealpha=1, shadow=**True**, borderpad=1,fontsize=**'xx-large'**)  
plt.title(**"Word Embedding Space with 20 most & least frequent words "**,size=20)  
vocab=list(model.wv.vocab)

**for** i, word **in** enumerate(vocab):  
 plt.annotate(word,xy=(neww\_X[i,0],neww\_X[i,1]))  
plt.savefig(**'most\_least\_words\_new\_stopwords\_top\_20.png'**, dpi=400, bbox\_inches=**'tight'**)  
*# plt.annotate(word,xy=(neww\_X[i+19,0],neww\_X[i+19,1]))****# ### Words with least frequency*****for** word **in** model.wv.vocab:  
 **if**(model.wv.vocab[word].count <180):  
 print((word, model.wv.vocab[word].count))

vocab1 = model.wv.index2entity[-20:] *#### Least 20 words*vocab1  
  
  
***# # WordCloud*****from** wordcloud **import** WordCloud  
wordcloud = WordCloud(background\_color=**"white"**, max\_words=5000, contour\_width=3, contour\_color=**'steelblue'**)  
long\_string = **','**.join(list(corpus))  
 *44*  
wordcloud.generate(long\_string)  
wordcloud.to\_image()  
  
 ***# ## Top 20 frequent words using BARGRAPH*****from** sklearn.feature\_extraction.text **import** CountVectorizer  
count\_vectorizer = CountVectorizer(stop\_words=**'english'**)

*# Fit and transform the processed titles*count\_data = count\_vectorizer.fit\_transform(corpus)  
  
**def** plot\_20\_most\_common\_words(count\_data, count\_vectorizer):  
 **import** matplotlib.pyplot **as** plt  
 words = count\_vectorizer.get\_feature\_names()  
 total\_counts = np.zeros(len(words))  
 **for** t **in** count\_data:  
 total\_counts+=t.toarray()[0]  
   
 count\_dict = (zip(words, total\_counts))  
 count\_dict = sorted(count\_dict, key=**lambda** x:x[1], reverse=**True**)[0:20]  
 words = [w[0] **for** w **in** count\_dict]  
 counts = [w[1] **for** w **in** count\_dict]  
 x\_pos = np.arange(len(words))   
   
 plt.figure(2, figsize=(15, 15/1.6180))  
 plt.subplot(title=**'20 most common words'**)  
 sns.set\_context(**"notebook"**, font\_scale=1.25, rc={**"lines.linewidth"**: 2.5})  
 sns.barplot(x\_pos, counts, palette=**'husl'**)  
 plt.xticks(x\_pos, words, rotation=90)   
 plt.xlabel(**'words'**)  
 plt.ylabel(**'counts'**)  
 plt.show()

***# Visualise the 20 most common words***plot\_20\_most\_common\_words(count\_data, count\_vectorizer)  
  
*45*

***# # Latent Dirichlet Allocation (LDA)***

**import** gensim.corpora **as** corpora  
dictionary = corpora.Dictionary(sent)  
*# Print dictionary*print(dictionary.token2id)  
  
*## Create Term document frequency (corpus)  
# Term Document Frequency*corpus = [dictionary.doc2bow(text) **for** text **in** sent]  
*# Print corpus for first document*print(corpus[0])  
  
  
[[(dictionary[id], freq) **for** id, freq **in** cp] **for** cp **in** corpus[:1]]  
  
  
*## BASELINE MODEL***import** gensim  
  
NUM\_TOPICS = 8  
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num\_topics = NUM\_TOPICS, id2word=dictionary,random\_state=100,passes=10)  
*# Saving trained model*ldamodel.save(**'LDA\_NYT'**)  
*# Loading trained model*ldamodel = gensim.models.ldamodel.LdaModel.load(**'LDA\_NYT'**)  
  
ldamodel.print\_topics(-1)

*# Compute Coherence*

from gensim.models import CoherenceModel Score

coherence\_model\_lda = CoherenceModel(model=ldamodel, texts=sent, dictionary=dictionary, coherence='c\_v')

*46*

coherence\_lda = coherence\_model\_lda.get\_coherence()

print('\nCoherence Score: ', coherence\_lda)  
*# supporting function*

def compute\_coherence\_values(corpus, dictionary, k, a, b):

lda\_model = gensim.models.LdaMulticore(corpus=corpus,

id2word=dictionary,

num\_topics=10,

random\_state=100,

chunksize=100,

passes=10,

alpha=a,

eta=b,

per\_word\_topics=True)

coherence\_model\_lda = CoherenceModel(model=ldamodel, texts=sent, dictionary=dictionary, coherence='c\_v')

return coherence\_model\_lda.get\_coherence()

*## Final Model*

lda\_model = gensim.models.LdaMulticore(corpus=corpus,

id2word=dictionary,

num\_topics=20,

random\_state=100,

chunksize=100,

passes=10,

alpha=0.01,

eta=0.9)

*47*

***### Visual Interpretation*****import** pyLDAvis  
**import** pyLDAvis.gensim   
  
pyLDAvis.enable\_notebook()  
plot = pyLDAvis.gensim.prepare(ldamodel, corpus, dictionary)  
*# Save pyLDA plot as html file*pyLDAvis.save\_html(plot, **'LDA\_NYT.html'**)  
plot

*48*