Text clustering, Sentiment analysis and Topic modeling with BBC news dataset

Hamed Ahmed[1] and Sugarbayar Enkhbayar[2]
[1] *University of Warsaw, Faculty of Economics*
h.hamedahmed@student.uw.edu.pl
[2] *University of Warsaw, Faculty of Economics*
s.enkhbayar@student.uw.edu.pl

Abstract: Main purpose of this project is to use text clustering, sentiment analysis and topic modeling methods on BBC news' text dataset

Keywords: text clustering, sentiment analysis, topic modeling

1. Introduction

a. Main purpose

The main goal of this project is to apply text clustering, sentiment analysis and topic modeling approaches to BBC news dataset using python. This project has 3 parts.

In the first part, the project focuses on applying text clustering techniques to the BBC news dataset. Two distinct clustering models, namely KMeans and DBSCAN, are employed to partition the dataset into meaningful clusters. To determine the optimal number of clusters, the silhouette score is utilized, providing a quantitative measure of the quality of clustering. This ensures a robust and well-defined grouping of news articles.

In the second part, the project explores topic modeling approaches, specifically Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). These methods aim to uncover latent topics within the dataset, offering insights into the overarching themes present in the BBC news articles.

In the third part, we focused on sentiment analysis. Leveraging Python's TextBlob library, the project delves into the emotional tone of the news articles. This involves categorizing each article as positive, negative, or neutral, providing a comprehensive understanding of the sentiment conveyed in the BBC news dataset.

b. Assumption of project

We made the following assumptions before making analysis using news data. Main goal of news is to inform and to educate readers, listeners or viewers. So all news agencies deliver the most important and hottest events in the world through news. 2 very important events have happened in the last 4 years. It was the Ukraine and Russian war, covid.

- Most of the news is likely to be related to war and covid
- Also, most of news is likely to be negative

2. Data

a. Description of dataset

We used the kaggle's BBC news dataset for our project. The data contains the following 5 columns title, pubDate, guide, link and description. We will mainly use the description column for our analysis. Because it contains text descriptions of each news.

Table: Description of data

Column	Description
title	It shows us title of each news
pubDate	It shows us published date
guide	It shows us link
link	It shows us link
description	Description of news

Table: Head of data

title	pubDate	guide	link	description
Ukraine: Angry Zelensky vows to punish Russian…	Mon, 07 Mar 2022 08:01:56 GMT	https://www.bbc.co.uk/news/world-europe-60638042	https://www.bbc.co.uk/news/world-europe-606380…	The Ukrainian president says the country will …
…	…	…	…	…

b. Preparation of data for modeling

Data preparation is one of the most important parts in data analysis. Because most data in the real world is so messy. Most data scientists spend most of their time cleaning data. We prepared our dataset through following steps for explanatory analysis and modeling.

- create following new variables workday, date, year, month, hour from published date variable
- converted month names to month numbers
- created new column that shows us number of word in each description

Text data requires more data cleaning process than numeric data. We prepared our text description through the following steps for text modeling.

- converted all letters to lower case
- removed all numbers from text
- removed special characters from text such as @.,?! so on
- applied lemmatization for text. We write the same word in many different ways. Using lemmatization , all words are converted to root words. Text models work well using lemmatization text.
- removed duplicate and missing data
- Removed white spaces

After the previous data cleaning process, data looks like the following table.

Table: Data after cleaning

Title	pubDate	description	workday	hour	date_fixed	year	month	hour_24	num_bword
Ukraine: Angry Zelensky vows to punish Russian…	Mon, 07 Mar 2022 08:01:56 GMT	ukrainian president say country forgive forget…	Mon	08:01:56	2022-03-07	2022	03	08	16

War in Ukraine: Taking cover in a town under a...	Sun, 06 Mar 2022 22:49:58 GMT	jeremy bowen frontline irpin residents come ru...	Sun	22:49:58	2022-03-06	2022	03	22	18
...	...	...	...	...	...	...	...	...	...

3. Explanatory analysis

We have 25396 BBC news dataset. 25386 news published in 2022 and 2023 (Fig: Number of data by year). Most data was collected from only these 2 years. So we decided to remove the other year's data. Then we did some exploratory data analysis.

- Around 2000 news articles are published monthly.
- But around 1000 news articles are published in January, February and March (Fig: Number of data by month). There are many celebration days and public holidays during these 3 months. It decreases the published number of news
- But the monthly published number of news increases from March to July
- More news articles are published in August, September and October. Because people go back to their job, school after summer holiday
- Most news is published in October. This shows that most people work a lot in October.

Figure: Number of data by year

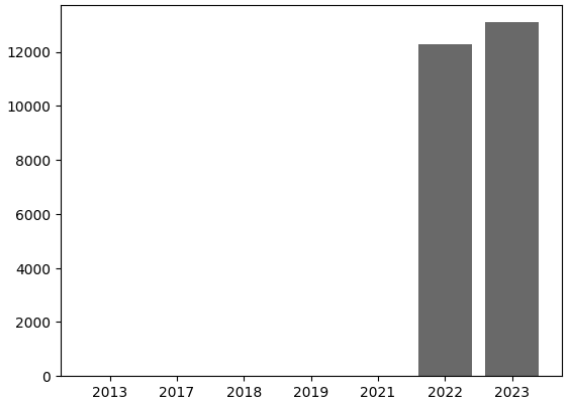
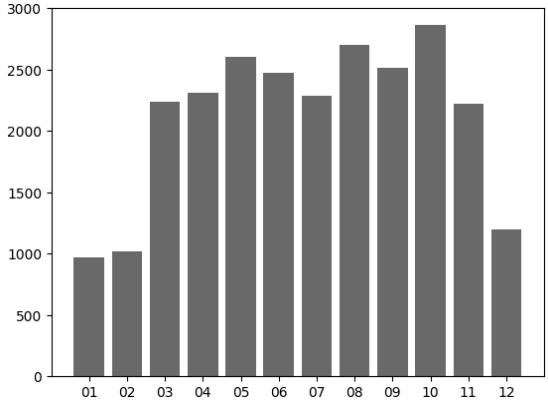


Figure: Number of data by month



Most news was published between 21:00 and 24:00 (Fig: Number of data by hour). Most description text contains 10 to 30 words without stop words. On average, one news article has around 18 words (Fig: Histogram of number of words).

Figure: Number of data by hour

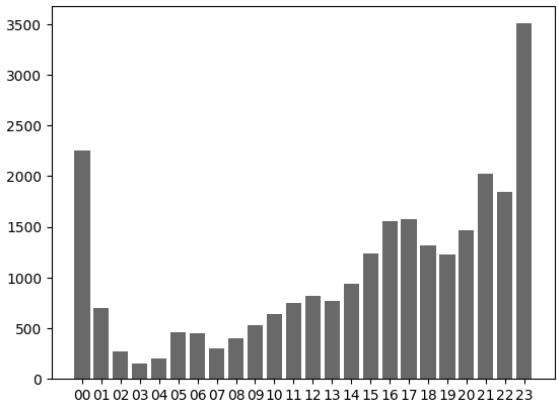
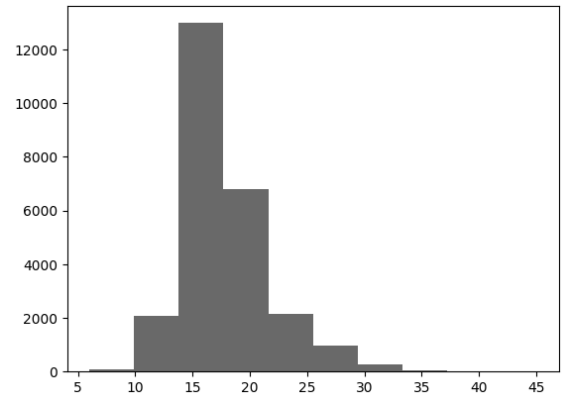


Figure: Histogram of number of word



Following figure shows us the most common words in the dataset. Bigger text indicates more words. Therefore, most of the news is likely to be related to the following words. For example: UK, England, Ukraine, World Cup, Government, Russia.

- BBC's headquarters is located in the UK. So most news related to the situation in the UK
- The FIFA WorldCup was held in 2022. Football is one of the most popular sports in the world. So much news about the World cup was published in 2022.
- War between Ukraine and Russia was one of the most important news in the world from 2022. On 24 February 2022, Russia invaded Ukraine in an escalation of the Russo-Ukrainian War that started in 2014.

Figure: WordCloud of dataset

4. Text clustering

Firstly, we used the TfidfVectorizer function to convert raw text data to a matrix of TF-IDF features. Then we created a sparse matrix using the vectorizer.fit_transform function where each row corresponds to text in data, and each column corresponds to a unique word in the vocabulary created by TfitVectorizer. After it, we trained 2 different clustering models (KMeans, DBSCAN) with silhouette scores.

KMeans

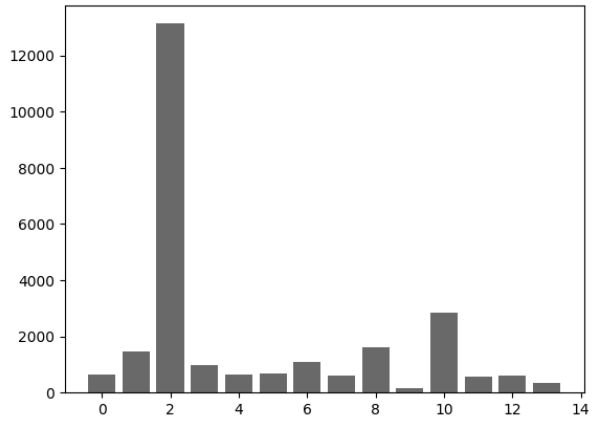
There is only one hyperparameter in the KMeans model, that is the number of clusters. We found the optimal number of clusters using silhouette score. The silhouette score is a metric used to calculate the goodness of a clustering technique. It measures how well-defined the clusters are in a given set of data. The silhouette score ranges from -1 to 1. Higher silhouette score will be better. It means a point is well-matched to its own cluster. Also, we considered the number of clusters between 2 to 15. Following table shows us that the optimal number of clusters is 14 (Table: Number of cluster and silhouette score). Because it has the highest silhouette score. So we used this optimal hyperparameter for the final KMeans model to divide texts into clusters.

Table: Number of cluster and silhouette score

Number of cluster	Silhouette score
2	0.002092
3	0.003035
4	0.004055
5	0.003684
6	0.004699
7	0.005205
8	0.005836
9	0.006057
10	0.00685
11	0.00688
12	0.00754
13	0.00793
14	**0.00849**
15	0.00841

Following figure shows us how much text was classified to each cluster. We can see most text data was clustered into cluster 2, 8 and 10. Cluster 2 especially has the most news.

Table: Number of dataset by cluster



We tried to determine characteristics of each cluster using the most frequent words. Following table shows us the top 10 words of each cluster. For example,

- cluster 3 is more likely to contain news about football. Because most words of cluster 3 are related to football team names.
- cluster 12 is more likely to contain news about the Russian and Ukrainian war. Because most words of cluster 12 are related to russia and ukraine.

Table: Top 10 words of each cluster

Num clus	Word
Cluster 1	uk, years, make, lead, year, police, home, come, time, win
Cluster 2	seven, past, days, closely, attention, whats, pay, selection, image, globe
Cluster 3	**league, manchester, unite, premier, city, champion, win, beat, score, liverpool**
Cluster 4	england, test, wales, ash, say, day, series, win, second, bank
Cluster 5	womens, cup, world, england, win, final, league, watch, game, australia
Cluster 6	people, say, kill, young, uk, die, help, thousands, children, home
Cluster 7	say, police, make, minister, president, uk, family, want, need, help
Cluster 8	new york, say, government, leader, plan, pm, years, minister, star
Cluster 9	leave, leader, say, labor, party, story, home, dead, minister, power
Cluster 10	live, cost, rise, energy, say, help, crisis, struggle, household, calculator
Cluster 11	world, cup, win, england, final, watch, qatar, beat, wales, france
Cluster 12	**ukraine war, russia, invasion, russian, say, russias, flee, eastern, putin**
Cluster 13	tell, bbc, story, people, say, news, make, want, years, help
Cluster 14	bbc, sport, speak, look, news, investigation, radio, reveal, access, year

DBSCAN

For the DBSCAN model, we have to find the optimal eps hyperparameter. It indicates maximum distance between two samples for one to be considered as in the neighborhood of the other. We used silhouette score to find it. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that doesn't require you to specify the number of clusters beforehand, unlike some other clustering algorithms such as k-means. Instead, DBSCAN identifies clusters based on the density of data points in the feature space. Following table shows us optimal eps is 1. Because it has the highest silhouette value. So we use it for the final DBSCAN model.

Table: Silhouette score of eps

Eps	Silhouette
0.1	-0.0304
0.325	-0.0308
0.55	-0.0318
0.775	-0.0355
1	-0.0197

Following table shows us the top 10 words of each cluster from the result of the DBSCAN model. Result of DBSCAN is very similar to the result of KMeans.

Table: Top 10 words of clusters

Num clus	Words
Cluster 1	uk, years, make, lead, year, police, home, come, time, win
Cluster 2	seven, past, days, closely, attention, whats, pay, selection, image, globe
Cluster 3	league, manchester, unite, premier, city, champion, win, beat, score, liverpool
Cluster 4	england, test, wales, ash, say, day, series, win, second, bank
Cluster 5	womens, cup, world, england, win, final, league, watch, game,australia

5. Topic modeling

Before topic modeling, we splitted each text into individual words. It allows us to represent texts as a list of words. Then we assigned a unique numerical ID to each unique word.Then we converted the texts into a document-term matrix using a bag-of-words representation. After it, we trained following 2 different models for topic modeling. (LSA and LDA)

LSA (Latent Semantic Analysis)

The Latent Semantic Analysis (LSA) model is a technique used in natural language processing and information retrieval to discover the relationships between words and topics within a collection of documents. It is a type of dimensionality reduction method. We used 10 numbers of topics that the model should find 10 underlying topics.

Following table shows us topics discovered by model. In other words, we can see the top words for each topic. Each topic is represented as a list of words, and the strength of association of each word with the topic is also provided. Previous numbers of each word indicate the strength of words.

Table: Result of LSA topic modeling

Topic	Top words of each topic
1	'0.822*"say" + 0.202*"world" + 0.172*"cup" + 0.140*"england" + 0.097*"win" + 0.092*"first" + 0.089*"people" + 0.084*"new" + 0.075*"take" + 0.074*"league"
2	'-0.493*"world" + 0.476*"say" + -0.426*"cup" + -0.211*"england" + -0.208*"win" + -0.163*"first" + -0.129*"womens" + -0.125*"league" + -0.119*"final" + -0.107*"bbc"
3	'-0.443*"world" + -0.336*"cup" + 0.296*"league" + 0.227*"first" + 0.208*"bbc" + -0.205*"say" + 0.187*"manchester" + 0.174*"people" + 0.140*"win" + 0.140*"new"
4	'0.430*"bbc" + -0.337*"league" + 0.331*"people" + -0.244*"win" + -0.229*"manchester" + 0.162*"tell" + -0.139*"champion" + -0.136*"first" + -0.136*"unite" + -0.136*"premier"
5	'0.491*"bbc" + -0.400*"first" + -0.359*"england" + 0.255*"league" + -0.251*"new" + 0.223*"manchester" + 0.161*"sport" + 0.142*"unite" + 0.138*"tell" + 0.133*"premier"
6	'0.569*"people" + -0.486*"england" + -0.422*"bbc" + 0.183*"world" + -0.135*"sport" + -0.121*"tell" + 0.113*"cup" + 0.104*"uk" + -0.102*"new" + 0.098*"ukraine"
7	'-0.550*"england" + 0.495*"first" + -0.270*"people" + 0.211*"win" + 0.207*"bbc" + -0.179*"manchester" + 0.157*"time" + -0.143*"league" + -0.113*"unite" + 0.111*"years"
8	'0.769*"new" + -0.334*"people" + -0.265*"england" + -0.213*"first" + -0.143*"bbc" + -0.121*"win" + 0.089*"world" + 0.082*"lead" + 0.079*"manchester" + -0.068*"time"

9	'0.605*"take" + -0.320*"people" + 0.263*"days" + -0.256*"new" + 0.238*"go" + 0.219*"seven" + 0.217*"past" + 0.184*"pay" + 0.180*"strike" + -0.134*"bbc"
10	'0.657*"win" + -0.409*"first" + 0.250*"one" + -0.213*"years" + -0.145*"cup" + -0.138*"time" + -0.138*"league" + -0.134*"manchester" + 0.117*"open" + 0.108*"ukraine"

LDA(Latent Dirichlet Allocation)

LDA is a probabilistic model that assumes each document in a collection is a mix of topics and that each word in the document is attributable to one of the document's topics. Following table shows us the top 10 words of each topic.

Table: Result of LDA topic modeling

Topic	Top words of each topic
1	'0.029*"say" + 0.020*"minister" + 0.016*"former" + 0.013*"party" + 0.013*"new" + 0.011*"leader" + 0.010*"prime" + 0.010*"labour" + 0.010*"bbc" + 0.009*"pm"
2	'0.012*"manchester" + 0.011*"city" + 0.010*"israeli" + 0.009*"england" + 0.008*"say" + 0.008*"football" + 0.007*"day" + 0.007*"boss" + 0.007*"top" + 0.006*"one"
3	'0.033*"say" + 0.012*"people" + 0.011*"find" + 0.011*"home" + 0.007*"die" + 0.007*"family" + 0.007*"star" + 0.007*"film" + 0.006*"hospital" + 0.006*"work"
4	'0.033*"world" + 0.031*"cup" + 0.020*"win" + 0.019*"league" + 0.015*"england" + 0.013*"womens" + 0.011*"unite" + 0.011*"final" + 0.011*"beat" + 0.010*"first"
5	'0.044*"bbc" + 0.013*"tell" + 0.011*"sport" + 0.011*"show" + 0.010*"us" + 0.009*"look" + 0.008*"reveal" + 0.008*"social" + 0.008*"media" + 0.008*"take"
6	'0.013*"first" + 0.012*"say" + 0.010*"years" + 0.009*"rugby" + 0.008*"storm" + 0.008*"star" + 0.007*"series" + 0.007*"public" + 0.006*"friends" + 0.006*"latest"
7	'0.017*"strike" + 0.015*"bbcs" + 0.009*"front" + 0.009*"king" + 0.008*"fall" + 0.008*"interest" + 0.008*"world" + 0.007*"paper" + 0.007*"editor" + 0.007*"rat"
8	'0.015*"pay" + 0.014*"say" + 0.011*"days" + 0.010*"go" + 0.009*"past" + 0.008*"seven" + 0.007*"attack" + 0.007*"dead" + 0.006*"ukraine" + 0.006*"president"
9	'0.025*"say" + 0.020*"people" + 0.014*"rise" + 0.009*"price" + 0.008*"uk" + 0.008*"cost" + 0.007*"find" + 0.007*"help" + 0.007*"government" + 0.007*"live"
10	'0.024*"police" + 0.021*"say" + 0.016*"israel" + 0.014*"attack" + 0.013*"man" + 0.012*"take" + 0.011*"gaza" + 0.009*"yearold" + 0.009*"officer" + 0.008*"arrest"

6. Sentiment analysis

We used the TextBlob library for this sentiment analysis. It is a simple NLP library that provides a consistent API for diving into common NLP tasks. There are 2 important features in this method. They are “polarity” and “subjectivity”.

Subjectivity values are between 0 and 1. Higher subjectivity value indicates that text contains personal opinion, emotion or judgment content.

Polarity values are between -1 and 1. Negative value indicates negative sentiment. Positive value indicates positive values. When we create sentiment types based on polarity value, we use the following rule.

- If polarity value is higher than 0, it will be “positive” sentiment
- If polarity value is lower than 0, it will be “negative” sentiment
- Otherwise, it will be “neutral” sentiment

Result of sentiment analysis shows the following table. We choose only one result from sentiment analysis for each type of sentiment and explain it. Following table shows the results of sentiment analysis.

- 1st text: Model gives us 1 polarity value, 0.3 subjectivity value based on this text. Polarity value is 1, so this text will be positive sentiment. As you can see, 1st text is positive news about Frank Sinatra
- 2nd text: Model gave us -1 polarity value, 1 subjectivity value based on this text. Polarity value is -1, so text will be negative sentiment. As you can see, 2nd text is negative news about alcohol
- 3rd text: Model gave us 0 polarity value, 0 subjectivity value based on this text. So text will be neutral sentiment

Table: Result of sentiment analysis

Text	Polarity	subjectivity	Sentiment
The singer described by Frank Sinatra as ""the best in the business""	1	0.3	**Positive**
Jit Chauhan, 48, was drinking a liter of whisky a day when his alcoholism was at its worst.	-1	1	**Negative**
Jeremy Bowen was on the frontline in Irpin, as residents came under Russian fire while trying to flee.	0	0	**Neutral**

Following table shows the number of data by sentiment type. 46% of data was classified neutral news. 35% of data was classified as positive. 18% of data was classified as negative news.

Table: Number of data by sentiment type

Sentiment type	Number of data	% of Total
Positive	8921	35%
Negative	4676	18%
Neutral	11789	46%

7. Summary

Our project embarked on the ambitious task of applying advanced text mining techniques, including topic modeling, text clustering, and sentiment analysis, to unravel patterns within a dataset dominated by unstructured data. With limited experience in handling unstructured data at the project's outset, our goal was to acquire practical skills and knowledge in navigating this challenging terrain.

Two primary assumptions framed our initial expectations: that the predominant topics within the dataset would be related to war and COVID, and that the overall sentiment of the news would lean towards the negative spectrum. The results of our analysis yielded intriguing findings, validating some assumptions while challenging others.

One of our assumptions held true, revealing a significant presence of topics related to the ongoing conflict between Ukraine and Russia. The prevalence of this theme underscores the power of text mining techniques in uncovering underlying patterns within unstructured datasets.

Contrary to expectations, the dataset contained limited news related to COVID. Exploring the reasons behind this observation could provide valuable insights into the dataset's composition, sources, or temporal characteristics. This unexpected finding opens avenues for further investigation and potential adjustments to the scope of future analyses.

While the assumption of predominantly negative news was not fully realized, the sentiment analysis results introduced an unexpected element—most of the data exhibited positive and neutral sentiments. Delving into the intricacies of sentiment analysis methodologies and understanding the contextual factors contributing to this positivity can shed light on the nuances of the dataset.

In retrospect, our project successfully achieved its primary purpose. We not only gained practical skills in handling unstructured data but also uncovered meaningful insights into the dataset's thematic landscape and sentiment distribution. The process of validating assumptions, discovering unexpected patterns, and critically evaluating results has deepened our understanding of text mining methodologies.

8. Appendix with source code and dataset

Appendix 1: Python code

```
# ! pip install nltk
# ! pip install spellchecker

#---------------------------------- Library ---------------------------------------
import pandas as pd
import numpy as np
from datetime import datetime
import re
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.stem.wordnet import WordNetLemmatizer
#---------------------------------- Data ---------------------------------------
from google.colab import drive
drive.mount('/content/gdrive')
df=pd.read_csv('bbc_news.csv')

#---------------------------------- Data Prepare ---------------------------------------
df=df[['title','pubDate','description']]
df['workday']=df['pubDate'].str.slice(0, 3)
df['date']=df['pubDate'].str.slice(4, 16)
df['hour']=df['pubDate'].str.slice(17, 25)

df["date"] = df["date"].apply(lambda x: x.replace(" Jan ", "-01-"))

for i,j in zip([' Jan ',' Feb ',' Mar ',' Apr ',' May ',' Jun ',' Jul ',' Aug ',' Sep ',' Oct ',' Nov ',' Dec
'],['-01-','-02-','-03-','-04-','-05-','-06-','-07-','-08-','-09-','-10-','-11-','-12-']):
   df["date"] = df["date"].apply(lambda x: x.replace(i, j))

df['date_fixed']=df['date'].str.slice(4,6)+'-'+df['date'].str.slice(0,3)+'-'+df['date'].str.slice(7,11)
df['date_fixed'] = pd.to_datetime(df['date_fixed'])
df['year'] = df['date_fixed'].dt.strftime('%Y')
df['month'] = df['date_fixed'].dt.strftime('%m')
df['hour_24']=df['hour'].str.slice(0, 2)
df = df.drop('date', axis=1)

df['num_word']=df["description"].apply(lambda n: len(n.split()))

#---------------------------------- Data cleaning ---------------------------------------
df['description'] = df['description'].str.lower() # lower case

df.description = df.description.replace('\d+', '', regex=True, inplace=False) # remove numbers
df.description = df.description.replace('[^\w\s]', '', regex=True, inplace=False) # remove special
characters such as .,%$?!
```

```
stop = stopwords.words('english')
df['description'] = df['description'].apply(lambda x: ' '.join([word for word in x.split() if word
not in (stop)])) # remove stop words such as that, what...

lemmatizer = WordNetLemmatizer() # Lemmatization or Stemming, prefix
def lemmatize_words(text):
  words = text.split()
  words = [lemmatizer.lemmatize(word,pos='v') for word in words]
  return ' '.join(words)
df['description'] = df['description'].apply(lemmatize_words)

df=df.drop_duplicates() # Remove duplicate
df['description'].isna().sum() # Remove missing data

df['description'] = df['description'].str.strip() # Remove whitespace

df.head()

#---------------------------------- Explanatory analysis ----------------------------------
len(df) # len
import matplotlib.pyplot as plt

df_hist=df.groupby(['year'])['year'].count().reset_index(name='num') # year
# plt.bar(df_hist['year'],df_hist['num'])

df_hist=df.groupby(['month'])['month'].count().reset_index(name='num') # year
# plt.bar(df_hist['month'],df_hist['num'])

df_hist=df.groupby(['hour_24'])['hour_24'].count().reset_index(name='num') # year
# plt.bar(df_hist['hour_24'],df_hist['num'])

plt.hist(df.num_word)
df['num_word'].describe() # stat word number

#---------------------------------- Outlier or Remove ----------------------------------
df=df[df['year']>="2022"]

#---------------------------------- Word cloud ----------------------------------
from wordcloud import WordCloud
words = ','.join(list(df['description'].values))
wordcloud = WordCloud(background_color="white", max_words=5000, contour_width=3,
contour_color='steelblue')
wordcloud.generate(words)
wordcloud.to_image()

from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
import seaborn as sns
from sklearn.metrics import silhouette_score
```

```
vectorizer = TfidfVectorizer(
   min_df = 5,
   max_df = 0.95,
   max_features = 8000,
   stop_words = 'english')
X = vectorizer.fit_transform(df['description'])
kmean = KMeans(n_clusters=14)
Y = kmean.fit_predict(X)
x_bar=pd.DataFrame(Y)
x_bar = x_bar.set_axis(['V'], axis=1)
x_bar=x_bar.groupby(['V'], as_index = False)['V'].count()
x_bar
plt.bar(x_bar.index,x_bar['V'])
feat = kmean.cluster_centers_.argsort()[:, ::-1]
facts = vectorizer.get_feature_names_out()
results={}
for i in range(14):
   list_ter = [ ]
   for ind in feat[i, :10]:
      list_ter.append(facts[ind])
   results[f'Cluster {i}'] = list_ter
df_clusters = pd.DataFrame.from_dict(results)
df_clusters

# ---------------- Text clustering ----------------
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
import seaborn as sns
from sklearn.metrics import silhouette_score

vectorizer = TfidfVectorizer(
   min_df = 5,
   max_df = 0.95,
   max_features = 8000,
   stop_words = 'english')
X = vectorizer.fit_transform(df['description'])

# KMeans
for n in range(2, 16):
  kmean = KMeans(n_clusters=n)
  Y = kmean.fit_predict(X)
  print(n,': ', silhouette_score(X, Y))
kmean = KMeans(n_clusters=14)
Y = kmean.fit_predict(X)
x_bar=pd.DataFrame(Y)
x_bar = x_bar.set_axis(['V'], axis=1)
x_bar=x_bar.groupby(['V'], as_index = False)['V'].count()
x_bar
plt.bar(x_bar.index,x_bar['V'])
```

```
feat = kmean.cluster_centers_.argsort()[:, ::-1]
facts = vectorizer.get_feature_names_out()
results={}
for i in range(14):
   list_ter = [ ]
   for ind in feat[i, :10]:
      list_ter.append(facts[ind])
   results[f'Cluster {i}'] = list_ter
df_clusters = pd.DataFrame.from_dict(results)
df_clusters

# DBSCAN
from sklearn.cluster import DBSCAN
eps = np.linspace(0.1, 1.0, 5)
for n in eps:
 dbs = DBSCAN(eps=n, min_samples=4)
 Y = dbs.fit_predict(X)
 print(n,': ', silhouette_score(X, Y)) #Silhouette Score:  A score of 1 means that the point is
well-matched to its own cluster, and a score of -1 means that it is better matched to another
cluster
dbs = DBSCAN(eps=1, min_samples=4)
Y = dbs.fit_predict(X)
feat = kmean.cluster_centers_.argsort()[:, ::-1]
facts = vectorizer.get_feature_names_out()
results={}
for i in range(5):
   list_ter = [ ]
   for ind in feat[i, :10]:
      list_ter.append(facts[ind])
   results[f'Cluster {i}'] = list_ter
df_clusters = pd.DataFrame.from_dict(results)
df_clusters

lda.print_topics()

# ---------------- Topic Modeling ----------------
df1 = df.description.values.tolist()
clean_corpus = [doc.split() for doc in df1]
from gensim import corpora
dictionary = corpora.Dictionary(clean_corpus)
doc_term_matrix = [dictionary.doc2bow(doc) for doc in clean_corpus]

# LSA model
from gensim.models import LsiModel
lsa = LsiModel(doc_term_matrix, num_topics=10, id2word = dictionary)
print(lsa.print_topics(num_topics=10, num_words=10))

# LDA model
from gensim.models import LdaModel
```

```
lda = LdaModel(doc_term_matrix, num_topics=10, id2word = dictionary)
print(lda.print_topics(num_topics=10, num_words=10))

# ---------------- Sentiment analysis ----------------
res_text=[]
res_res=[]
res_pol=[]
res_sub=[]
from textblob import TextBlob
for text in df1:
  sent = TextBlob(text)
  pol = sent.sentiment.polarity
  sub = sent.sentiment.subjectivity
  if pol > 0:
    res = "Positive"
  elif pol < 0:
    res = "Negative"
  else:
    res = "Neutral"
  res_text.append(text)
  res_res.append(res)
  res_pol.append(pol)
  res_sub.append(sub)

data = {'text': res_text,
    'res': res_res,
    'pol': res_pol,
    'sub': res_sub}
result_sentiment = pd.DataFrame(data)
print(result_sentiment)

result_sentiment.groupby(['res'])['res'].count()
```