COMPARE NEWS ARTICLES RELATED TO CLIMATE CHANGE

BIA660-A – Web Mining

Prof Rong Liu

Team Term Project

(Aditya Gaikwad

Hanish Pallapothu

Moneka Ruhiil

Rishab Agrawal)

Stevens Institute of Technology

(17 December 2022)

Table of Contents

Abstract

The project's goal is to compare news articles related to climate change from various countries. Every country today faces problems with climate change; few countries are concerned about global warming, while others are concerned about monsoons and rainfall.

Various News Channels have different opinions regarding climate change, which is the primary source for data comparison throughout the project. The idea is to scrape news articles data from various countries and analyze the opinion of different authors. The project focuses on clustering the data using various Unsupervised algorithms in Machine Learning. The algorithms are measured based on both intrinsic and extrinsic evaluation methods.

Introduction

Contemporary climate change includes both global warming caused by humans, and its impacts on Earth's weather patterns. There have been previous periods of climate change, but the current changes are more rapid than any known events in Earth's history. It is one of those critical issues that don’t receive enough attention from the AI community. The main reason why machine learning developers and data scientists are building so few climate models is that climate change is hard to analyze. We try to cluster news of two countries namely India and America; then try to compare the clustering algorithm’s performance as well as using the best possible parameters and vectorizers. We try to answer the following questions: -

How well can some list of algorithms cluster news data?

What can be the best approaches for clustering data and how to evaluate performance of the approaches?

Doing a general polarity-based sentiment analysis of the news of two countries to compare the result and understand a general sentiment of them.

Data Collection

We scrapped the data from 4 reputed media websites which are 3 websites from US and 1 from India: -

- Hindustan Times
- NBC
- New York Times
- BBC news

Newspapers cover every topic concerning to public including topics like business, the world, sports, technology etc. Of all we focused on the topic **"climate change"** on which we will be performing sentiment analysis further down the model. We used the selenium library to scrape all the data. We scraped all the texts of climate change including title and description. There were also comments from users as well.

Other libraries used for web scraping:

Requests: It is an efficient HTTP library used for accessing web pages.

Urlib3: It is used for retrieving data from URLs.

Selenium: It is an open-source automated testing suite for web applications across different browsers and platforms.

text
WASHINGTON — The new infrastructure law signed by President Biden includes almost $50 billion to protect communities against climate change, the largest such investment in United States history and a recognition that the effects of warming are c
WADING RIVER, N.Y. — If Bill Jacobs were a petty man, or a less religious one, he might look through the thicket of flowers, bushes and brambles that encircle his home and see enemies all around. For to the North, and to the South, and to the West
CARNAÚBA DOS DANTAS, Brazil — The land has sustained the Dantas family for more than 150 years, bearing fields of cotton, beanstalks up to a grown man's hip and, when it rained enough, a river that led to a waterfall. But on a recent day, with te
Royal Dutch Shell said Thursday that it had decided not to invest in a British oil development off the coast of Scotland that has become a test of the government's environmental credentials. The field, known as Cambo, is in deep water northwest of th
Brynn Kimber, a research scientist at the University of Washington who works in the National Oceanic and Atmospheric Administration's Marine Mammal Laboratory, has spent a lot of time analyzing audio data recorded in the icy waters north of Alask
This personal reflection is part of a series called Turning Points, in which writers explore what critical moments from this year might mean for the year ahead. You can read more by visiting the Turning Points series page. Turning Point: The United Nat
To hear more audio stories from publications like The New York Times, download Audm for iPhone or Android. The day after the storm swallowed her neighborhood, Nancy Ortiz woke before dawn to buy ice. It was 2012, and Hurricane Sandy had recl
Hindsight is a series from the Headway team looking back at predictions and promises from the past. As the 2009 global climate summit in Copenhagen approached, the European Union raced to announce an ambitious target for reducing greenhouse
Hindsight is a series from the Headway team looking back at predictions and promises from the past. When a shopper in New York, say, plucks a Milky Way bar from a grocery store shelf, that shopper becomes the final link in a long chain that might
Advocates descended on the streets of Glasgow last month, pressing banks and other financial institutions at 26th United Nations climate summit to be more responsible stewards of the climate. But a bank based just 50 miles east of the Scottish city
GENEVA — When the United Nations made its last appeal for humanitarian aid funding before the pandemic, it asked donors for about $29 billion. But in the past year alone, there has been a huge jump in the number of people needing help. And so t
Back-to-back years of little precipitation in the Indian Ocean nation of Madagascar have ruined harvests and caused hundreds of thousands of people to face uncertainty about their next meals. Aid groups say the situation there is nearing a humanita
We're also covering a political struggle that could shape the future of clean energy, a U.S. government report on oil drilling that barely mentions climate change, and albatross divorces. By Christopher Flavelle The largest museum complex in the worl
Washed-out roads and destroyed bridges are just some of the devastation afflicting residents of the western province and could signal what climate change will bring in the future. By Ian Austen Photographs by Ian Willms PRINCETON, British Columb
GLASGOW — Inver restaurant is but a speck on the longest sea loch in Scotland. From its windows, a diner can see the remnants of a 15th-century castle and the rolling hills of the Highlands, but the breakout star is not the view. It's a meaty halibut
KASULO, Democratic Republic of Congo — A man in a pinstripe suit with a red pocket square walked around the edge of a giant pit one April afternoon where hundreds of workers often toil in flip-flops, burrowing deep into the ground with shovels and
PARIS — European countries desperate for a long-term and reliable source of energy to help reach ambitious climate goals are turning to an answer that caused earlier generations to shudder: nuclear power. Poland wants a fleet of smaller nuclear po
When a wildfire plows through a forest, life underground changes, too. Death comes for many microorganisms. But, like trees, some microbes are adapted to fire. Certain fungi are known as pyrophilous, or "fire-loving." After a fire, pyrophilous fungi
Arctic. Atlantic. Long ago, the two oceans existed in harmony, with warm and salty Atlantic waters gently flowing into the Arctic. The layered nature of the Arctic — sea ice on top, cool freshwater in the middle and warm, salty water at the bottom — I
For two weeks the Iranian government tolerated growing protests over scarce water supplies in the central Iranian city of Isfahan, watching them grow as restaurants served demonstrators free soup and barbers offered free haircuts. State television
BERLIN — In the middle of Germany's election campaign, almost 200 people died in extreme floods in Germany. Four months later, the fight against climate change has become the central theme of the new post-Merkel government. Most roofs will be
If you think what climate change portends for America is scary, wait until you hear about Australia. That's the gist of "Burning," which focuses on that country's sadly familiar experiences with warming temperatures: terrifying wildfires, drill-baby-drill
The redoubtable Rivian R1T, the first crusher in a coming wave of electric pickup trucks, can soar unscathed over gnarly boulders, hitch an 11,000-pound load and scorch 60 m.p.h. in about 3.5 seconds. The truck brings everything and the kitchen sir
We're also covering biodiversity collapse, the biggest climate legislation in United States history, how traffic roundabouts fight climate change, and more. By Dionne Searcey Demand for electric vehicles, a vital part of the push against climate change
Maya Lin's acclaimed "Ghost Forest" — her installation at Madison Square Park in New York — was being carved up, and the artist couldn't have been happier: A group of teenagers had seen the harvesting of the wood on Nov. 19 and were sawing it c
WASHINGTON — Private lenders will play a greater role in funding disaster recovery under a program announced Tuesday, moving to fill a gap left by sluggish federal aid programs that take years to get money to victims of floods, wildfires and other
Este otoño, cuando los científicos planeaban una expedición en México para contar los ejemplares de uno los animales más amenazados del mundo, una tímida marsopa llamada vaquita, temieron que no quedara ninguna por encontrar. El último recu
To hear more audio stories from publications like The New York Times, download Audm for iPhone or Android. As scientists planned an expedition in Mexico this fall to count one of the world's most endangered animals, a shy porpoise called a vaquita,

Figure 1 Sample screenshot of scrapped data of the American newspaper

Methodology

1) Selection of an online news article.

2)Extraction of sentences from the news articles. Sentences can be simple, compound, complex, and compound complex.

3) Search for positive words, phrases, or clauses in those sentences and find their polarities.

4) Combining the polarities of all sentences to get the final polarity of the news article. 91% accurate results were collected for the classification of news articles.

We investigate the relationship between mass media and public opinion by analyzing news articles for sentiment and topic detection. We apply this methodology to a specific case study, which is recent climate change, by analyzing news articles that contain the relevant keywords, such as CO2, global warming, and so on. We look at the number of times a keyword is mentioned and the evolution of the relative coverage of a set of topics within the same article.

Data Preprocessing

After collecting the data, we processed the data for cleaning before starting with building and training the model on it. This is one the most critical step in building models. We utilized EDA analysis to assist us in examining and comprehending the data, as well as extracting insights and key characteristics. We conducted exploratory analysis in the following manner: - We preprocessed the news articles included in the collection. Preprocessing is a critical step in cleaning text and reducing inconsistencies in it so that the cleansed data can be used more successfully in text mining or sentiment analysis tasks after it has been cleaned.

These cases we taken care of as part of data cleaning:

1. Checking for null, missing and duplicates values.
2. Then removed stopwords, tokenized it.
3. Removed all the other trailing, spaces and noisy using regex.
4. We removed punctuations and remove characters less than length 1
5. Convert everything to lowercase
6. Lemmatize the tokens, that is taking the root word out of different words.

Vectorization

The amount of textual data available is enormous, and the challenge is that it must be represented in a manner that can be used mathematically to solve a problem. To put it another way, we must obtain an integer representation of a word. This problem can be solved in a variety of ways, from basic to sophisticated. Vectorization is a process of converting text into numerical entries of a matrix form depending on the algorithm or logic behind the vectorization process. We have use 3 different methods of vectorization in our work about which we are going to describe in a few lines: -

1. Bag of Words or BOW. The idea of the BOW model is very simple. We would like to find the fraction of specific words that are strong indicators of positive or negative sentiment. For example, a review contains more positive words like “great”, “fantastic” would be considered more positive than a review with words like “so-so”. Depending on this we give certain numbers to the words so we can represent data in a numerical manner. We have used them as they interpret data in an easy way and work fast , better BOW isn’t necessarily the best model for machine learning every time.

2. That’s the reason we took TF-IDF (Term frequency – Inverse Document frequency) as one of the vectorizers. (TF) It is a measure of how frequently a term, t, appears in a document. IDF is a measure of how important a term is. The number of times a term appears in a particular document is represented by the symbol tf. As a result, it is specific to a document. The degree of frequency (df) of a term across the entire corpus of documents is a measure of its prevalence or rarity. So the important thing to remember is that it's consistent throughout all of the documents. If the word is common and appears in many documents, the idf value (normalized) will be close to zero, or close to one if the word is rare. It is the product of a term's two tf values that gives it its tf-idf value in a document. The higher the value, the more relevant the term is in that particular document. IDF is defined as the log of the ratio of number of documents to the number of documents with a specific term. So TF-IDF contains information about more important and less important words as well.

3. Third vectorizer used in this work is, Word2Vec. Word2vec is a collection of linked models for generating so-called word embeddings. These are shallow, two-layer neural networks that have been taught to recreate word linguistic contexts. Word2vec models may be used to

map each word to a vector of several hundred elements that represents that word's relationship to other words after training. The hidden layer of the neural network is represented by this vector. To produce neural word embeddings, Word2vec uses either skip-grams or a continuous bag of words (CBOW).

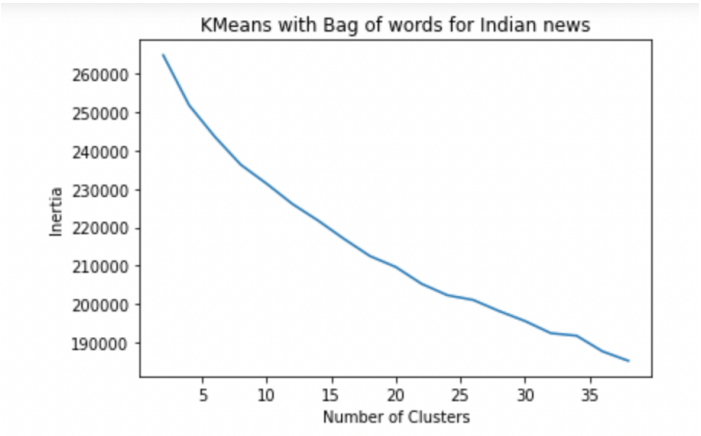
The purpose of using these different vectorizing approaches is to study the average number of clusters being produced for the same dataset with different clustering algorithms. Moving ahead in our work we are having Indian news and American news.

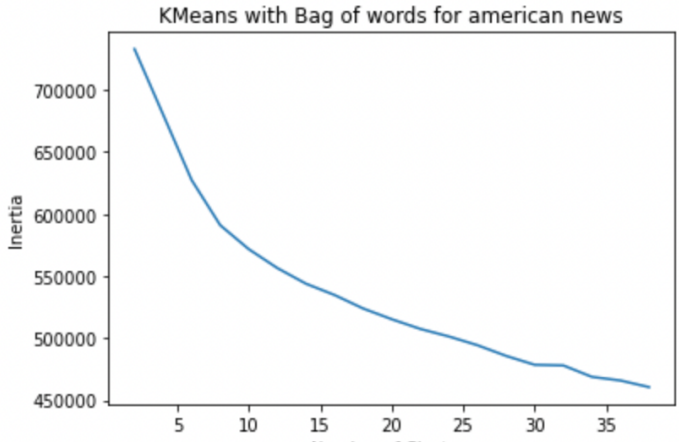
Clustering

K-Means

We use K-means algorithm for clustering all the three vector approaches for Indian and American news. K-means clustering is an unsupervised learning strategy that is utilized when we don't have labeled data, which we don't have in this scenario (means, without defined categories or groups). The purpose of this method is to locate groups in the data, with the variable K representing the number of groups. The data was grouped based on high similarity points clustering together and low similarity points clustering separately. Inertia is one of the important parameters in K-means, It measures how well a dataset was clustered by K-Means. It is calculated by measuring the distance between each data point and its centroid, squaring this distance, and summing these squares across one cluster.

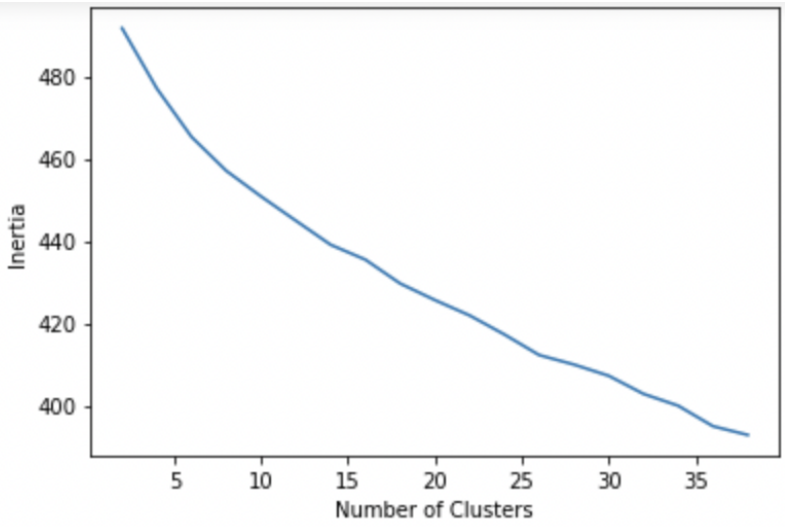
We calculate the optimum k (number of clusters) using the elbow method. Here we can see below the plots of elbow method for Indian news as well as American news for the Bag of Words vectorizer.





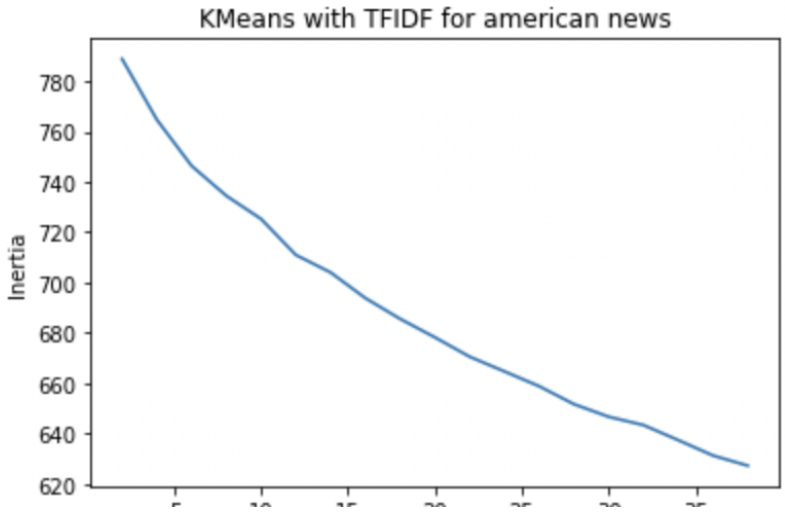
We get value of k as 8 for both American news and Indian news according to this for bag of words vectorizer.

TF-IDF vectorizer



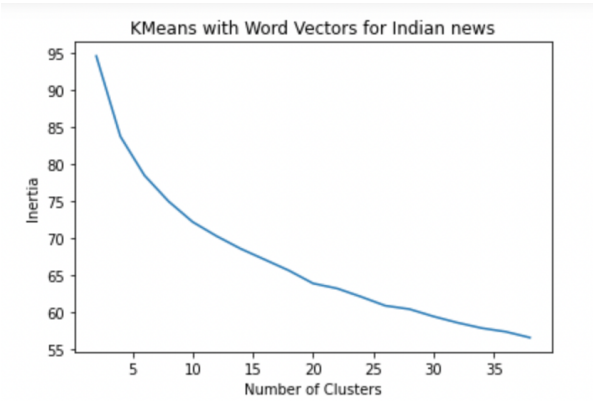
In [42]:
```
import matplotlib.pyplot as plt

plt.plot(n,inertia1)
plt.title("KMeans with TFIDF for american news")
plt.xlabel("Number of Clusters")
plt.ylabel("Inertia")
```

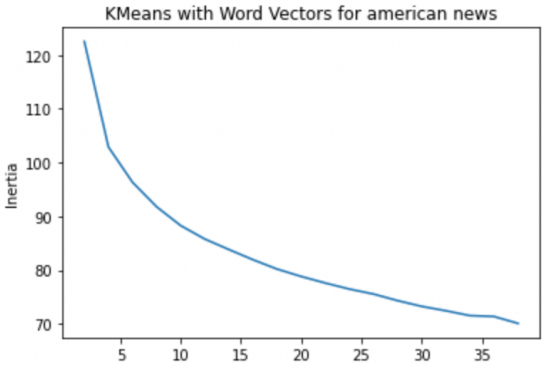
Out[42]:
```
Text(0, 0.5, 'Inertia')
```



The same way we get clusters as 9 and 6 for Indian news and American news in case of TF-IDF vectorizer.

Word2Vec





We get clusters as 4 and 4 for Indian news and American news in case of Word2Vec.

We then use the following cluster values as per elbow plots to get our prediction by fitting the K-means model with the appropriate cluster values and print the top 10 words of every cluster or the wordclouds. We can see it clearly in the image below. Then we used the Gaussian mixture model, this is majorly used for data that has higher variance as the shape of the data points is like a gaussian distribution curve so it can form shapes from circular to elliptical. So, this curve gives us a lot of flexibility in terms of the cluster shape.

```
cluster1_labels = clusterer1.fit_predict(bow1)

In [37]: df["labels"] = cluster_labels
         # f["labels"] = cluster1_labels
         idx_to_word = {values:key for key,values in count.vocabulary_.items(

         vocab_idx = np.argsort(clusterer.cluster_centers_, axis=1)[:,::-1][:

         for i in range(8):
             print("="*50)
             print("Cluster ",i+1," talks about :")
             print()

             a = [print(idx_to_word[x]) for x in vocab_idx[i]]
==================================================
Cluster  1  talks about :

ice
sea
temperature
arctic
ocean
year
level
rise
warming
scientist
==================================================
Cluster  2  talks about :

year
world
also
one

In [38]: cluster1_labels.shape
```

Gaussian Mixture Model

There are various metrics to evaluate performance of GMM model, like AIC, BIC, R square etc. The Akaike Information Criterion, or AIC for short, is a method for scoring and selecting a model. It is named for the developer of the method, Hirotugu Akaike, and may be shown to have a basis in information theory and frequentist-based inference. The Bayesian Information Criterion, or BIC for short, is a method for scoring and selecting a model. It is named for the field of study from which it was derived: Bayesian probability and inference. Like AIC, it is appropriate for models fit under the maximum likelihood estimation framework. Compared to the BIC method (below), the AIC statistic penalizes complex models less, meaning that it may put more emphasis on model performance on the training dataset, and, in turn, select more complex models. Though AIC and BIC are both Maximum Likelihood estimate driven and penalize free parameters in an effort to combat overfitting, they do so in ways that result in significantly different behavior.

```
    for n_components in n_components_range:

        # Fit a Gaussian mixture with EM
        gmm = mixture.GaussianMixture(n_components=n_components,
                    covariance_type=cv_type, random_state=42)
        gmm1 = mixture.GaussianMixture(n_components=n_components,
                    covariance_type=cv_type, random_state=42)
        gmm.fit(bow_vectors.toarray())
        gmm1.fit(bow1.toarray())
        bic = gmm.bic(bow_vectors.toarray())  # get Model BIC
        bic1 = gmm1.bic(bow1.toarray())
        if bic < lowest_bic:  # save the model with lowest BIC sofar
            lowest_bic = bic
            best_gmm = gmm
        if bic1 < lowest_bic1:
            lowest_bic1 = bic1
            best_gmm1 = gmm1
```

```
lowest_bic
best_gmm
```

```
GaussianMixture(covariance_type='diag', n_components=9, random_state=
```

```
lowest_bic1
best_gmm1
```

```
GaussianMixture(covariance_type='diag', n_components=7, random_state=
```

```
clusterer = GaussianMixture(9)

cluster_labels = clusterer.fit_predict(bow_vectors.toarray())

clusterer1 = GaussianMixture(7)

cluster1_labels = clusterer1.fit_predict(bow1.toarray())
```

GMM Results- BIC has the danger of underfitting with too much samples coming and AIC has risk of overfitting, but in our case we have preferred using BIC as it takes into consideration number of samples which is important for our clustering task. We have varied the cluster size from 2 to 10 and covariance types tied , spherical , diagonal and found the model have the least BIC.

```python
vocab_idx = np.argsort(clusterer.means_, axis=1)[:,::-1][:,:10]


#picking  5 important clusters based on weights
for i in clusterer.weights_.argsort()[::-1][:5]:
    print("="*50)
    print("Cluster ",i," talks about :")
    print()

    a = [print(idx_to_word[x]) for x in vocab_idx[i]]
```

```
==================================================
Cluster  7  talks about :

year
state
also
area
study
people
forest
government
global
one
==================================================
Cluster  4  talks about :

world
year
country
```

```
Cluster  5  talks about :

year
people
world
would
also
cop
government
country
one
emission
==================================================
Cluster  1  talks about :

emission
say
year
carbon
country
government
world
energy
gas
new
==================================================
Cluster  0  talks about :

would
new
year
state
community
people
one
time
company
```

We found the best BIC having 9 number of clusters for Indian news news and 7 for American news in case of Bag of Words .You can see the result in the above image. We fit the best found model on the data to predict the labels. After that we display the top 5 clusters out of the clusters formed and the top 10 words of the cluster.

```
        gmm1.fit(tf_idf1.toarray())
        bic = gmm.bic(tf_idf_vectors.toarray())  # get Model BIC
        bic1 = gmm1.bic(tf_idf1.toarray())
        if bic < lowest_bic:  # save the model with lowest BIC sofar
            lowest_bic = bic
            best_gmm = gmm
        if bic1 < lowest_bic1:
            lowest_bic1 = bic1
            best_gmm1 = gmm1
```

In [71]:
```
lowest_bic
best_gmm
```

Out[71]: GaussianMixture(covariance_type='diag', n_components=9, random_state=42)

In [72]:
```
lowest_bic1
best_gmm1
```

Out[72]: GaussianMixture(covariance_type='diag', n_components=9, random_state=42)

In [73]:
```
clusterer = GaussianMixture(9)

cluster_labels = clusterer.fit_predict(tf_idf_vectors.toarray())

clusterer1 = GaussianMixture(9)

cluster1_labels = clusterer1.fit_predict(tf_idf1.toarray())
```

In [74]:
```
df["labels"] = cluster_labels

idx_to_word = {values:key for key,values in vectorizer.vocabulary_.items()}

vocab_idx = np.argsort(clusterer.means_, axis=1)[:,::-1][:,:10]

for i in clusterer.weights_.argsort():
    print("="*50)
    print("Cluster ",i," talks about :")
    print()

    a = [print(idx_to_word[x]) for x in vocab_idx[i]]
```

```
paris
joe
==================================================
Cluster  3  talks about :

ice
glacier
sea
arctic
antarctica
snow
region
sheet
study
level
==================================================
Cluster  0  talks about :

thunberg
greta
```

```
Cluster  1  talks about :

water
temperature
year
heat
ice
flood
people
extreme
report
weather
==================================================
Cluster  3  talks about :

country
emission
coal
china
global
agreement
nation
world
fossil
fuel
==================================================
Cluster  0  talks about :

people
school
say
young
cop
plastic
food
world
```

We found the best BIC having 9 number of clusters for Indian news and 9 for American news in case of TF-IDF vectorizer .You can see the result in the above image. We fit the best found model on the data to predict the labels. After that we display the top 5 clusters out of the clusters formed and the top 10 words of the cluster.

```
                    covariance_type=cv_type, random_state=42)
        gmm.fit(word_vectors)
        gmm1.fit(word_vectors1)
        bic = gmm.bic(word_vectors)  # get Model BIC
        bic1 = gmm1.bic(word_vectors1)
        if bic < lowest_bic:  # save the model with lowest BIC so
            lowest_bic = bic
            best_gmm = gmm
        if bic1 < lowest_bic1:
            lowest_bic1 = bic1
            best_gmm1 = gmm1
```

```
lowest_bic
best_gmm
```

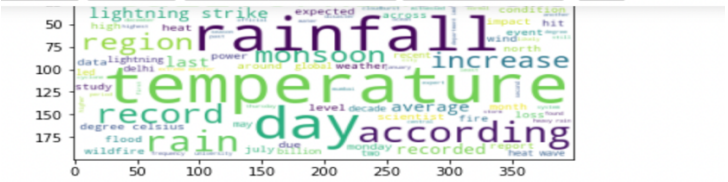
```
GaussianMixture(covariance_type='spherical', n_components=9, randc
```

```
lowest_bic1
best_gmm1
```

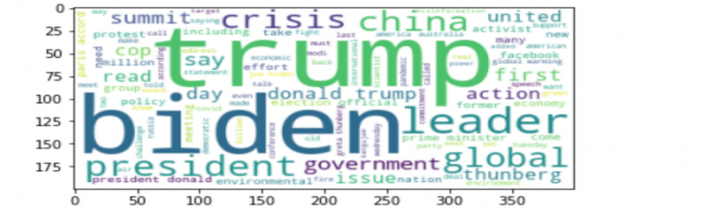
```
GaussianMixture(covariance_type='tied', n_components=4, random_sta
```

```
clusterer = GaussianMixture(9)

cluster_labels = clusterer.fit_predict(word_vectors)

clusterer1 = GaussianMixture(4)

cluster1_labels = clusterer1.fit_predict(word_vectors1)
```

```
clusterer = GaussianMixture(9)

cluster_labels = clusterer.fit_predict(word_vectors)
```




```
==================================================
Cluster  3  talks about :
```




```
==================================================
Cluster  4  talks about :
```

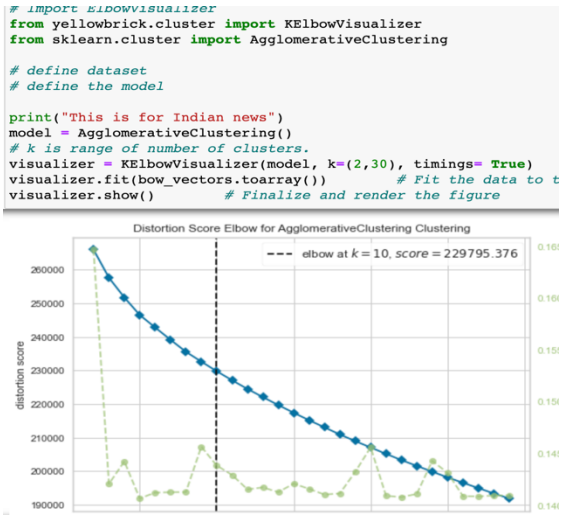
We found the best BIC having 9 number of clusters for Indian news and 4 for American news in case of Word2Vec vectorizer .You can see the result in the above image. We fit the best found model on the data to predict the labels. After that we display the top 5 clusters out of the clusters formed and the top 10 words of the cluster.

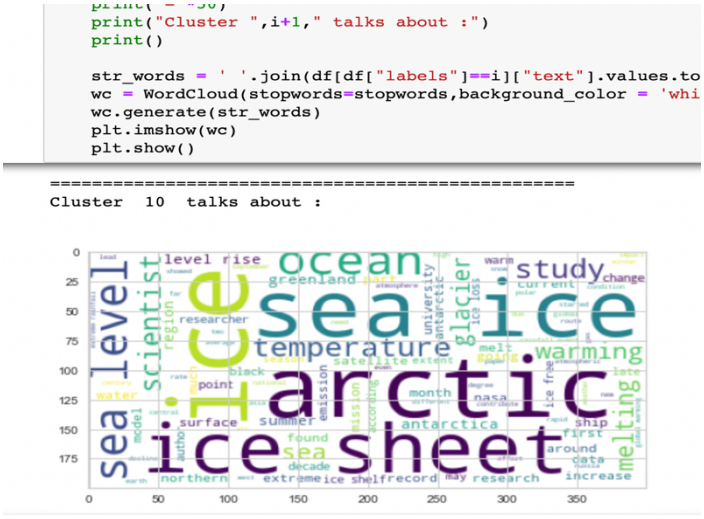
Agglomerative Clustering

Algorithm for clustering data in a hierarchical structure Agglomerative Clustering is a type of hierarchical clustering technique that uses a large number of clusters. Unsupervised machine learning is used to divide the population into clusters, with data points in the same cluster being more similar and data points in other clusters being more dissimilar than data points in other clusters. In comparison to the unstructured set of clusters returned by flat clustering, this structure provides more information. We are not required to specify the number of clusters in advance with this clustering algorithm. Bottom-up algorithms treat each piece of data as a singleton cluster at the outset, and then agglomerate pairs of clusters one by one until all clusters have been merged into a single cluster that contains all of the data, which is called a singleton cluster. The points in a cluster are closer to one another. The points in the various clusters are separated by a large distance.

Steps in Agglomerative Clustering: At first, each data point is in its own cluster. In step 2 we combine the two closest clusters to produce a single cluster. Step 2 should be repeated until you get the required number of clusters.

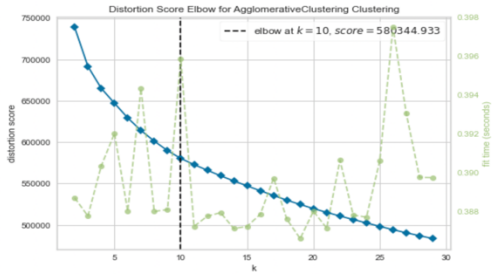
For agglomerative clustering we have created an elbow graph before clustering using cluster values from 2 to 30 using a visualiser from yellowbricks package.





```
print("This is for American news")
model = AgglomerativeClustering()
# k is range of number of clusters.
visualizer = KElbowVisualizer(model, k=(2,30), timings= True)
visualizer.fit(bowl.toarray())        # Fit the data to the visualizer
visualizer.show()        # Finalize and render the figure
```

This is for American news




```
#picking  5 important clusters based on weights
for i in range(10):
    print("="*50)
    print("Cluster ",i+1," talks about :")
    print()

    str_words = ' '.join(f[f["labels"]==i]["text"].values.tolist())
    wc = WordCloud(stopwords=stopwords,background_color = 'white',ma
    wc.generate(str_words)
    plt.imshow(wc)
    plt.show()
```

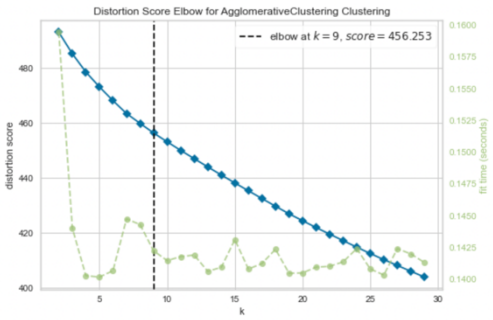
```
Cluster  2  talks about :
```


```
==================================================
```

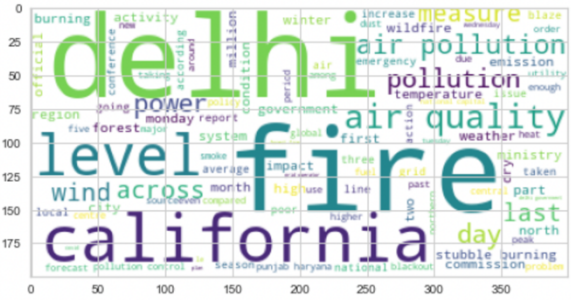
Using agglomerative clustering we found 10 number of clusters for Indian news and 10 for American news in case of BOW vectorizer. The above images are for Indian news and American news for some cluster images as well as elbow method graph.

```
# k is range of number of clusters.
visualizer = KElbowVisualizer(model, k=(2,30), timings= True)
visualizer.fit(tf_idf_vectors.toarray())        # Fit the data to the v
visualizer.show()        # Finalize and render the figure
```

This is for Indian news

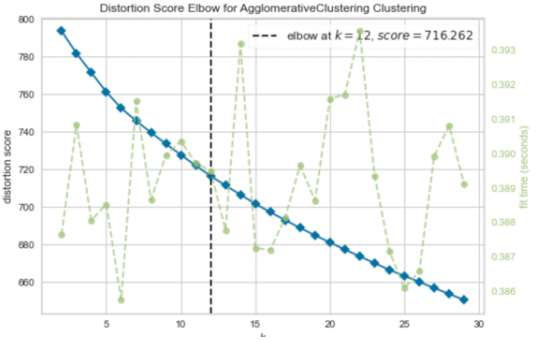



```
==================================================
Cluster  1  talks about :
```




```
visualizer = KElbowVisualizer(model, k=(2,30), timings= True)
visualizer.fit(tf_idf1.toarray())        # Fit the data to the vi
visualizer.show()        # Finalize and render the figure
```

This is for American news



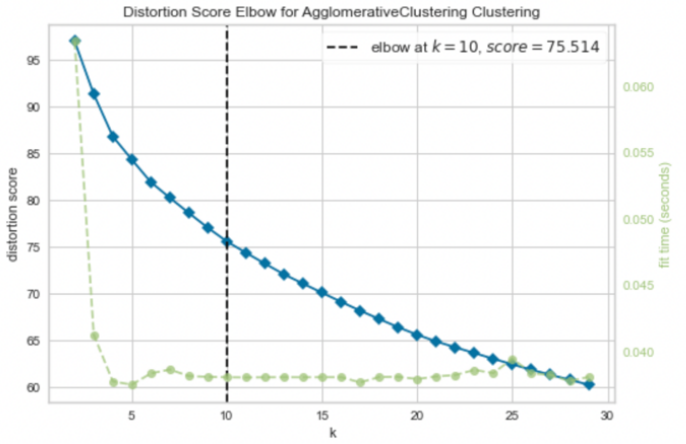

```
    str_words = ' '.join(f[f["labels"]==i]["text"].values.tolis
    wc = WordCloud(stopwords=stopwords,background_color = 'whit
    wc.generate(str_words)
    plt.imshow(wc)
    plt.show()
```

```
==================================================
Cluster  10  talks about :
```

Using agglomerative clustering we found 9 number of clusters for Indian news and 10 for American news in case of TF-IDF vectorizer. The above images are for Indian news and American news for some cluster images as well as elbow method graph.

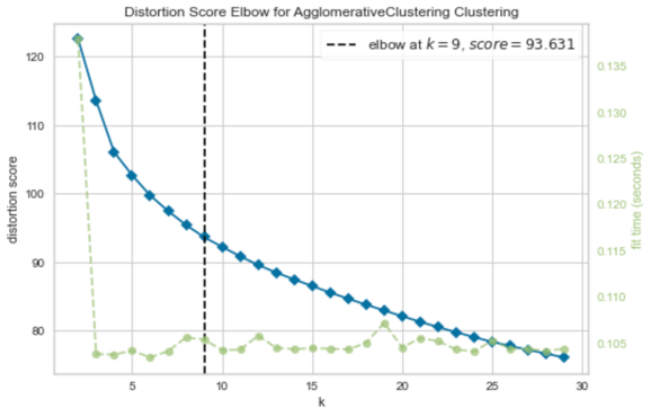
```
visualizer.show()        # Finalize and render the figure
```

```
This is for Indian news
```




```
print("This is for American news")
model = AgglomerativeClustering()
# k is range of number of clusters.
visualizer = KElbowVisualizer(model, k=(2,30), timings= True)
visualizer.fit(word_vectors1)        # Fit the data to the visualizer
visualizer.show()        # Finalize and render the figure
```

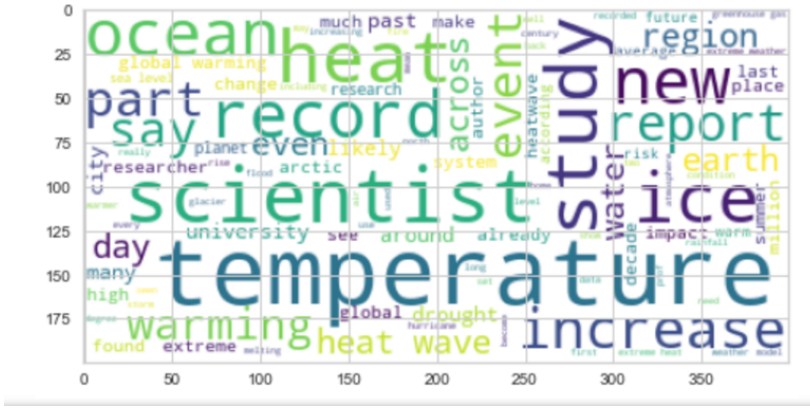
```
This is for American news
```




```
    wc.generate(str_words)
    plt.imshow(wc)
    plt.show()
```

```
==================================================
Cluster  8  talks about :
```

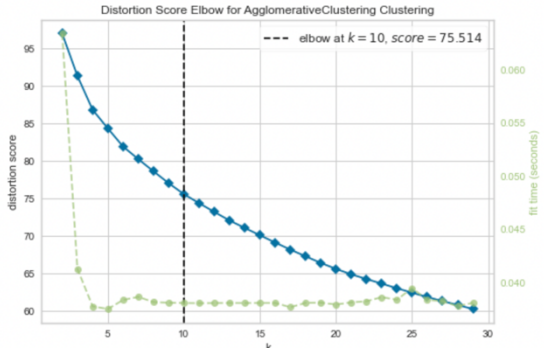

```
==================================================
Cluster  9  talks about :
```



Using agglomerative clustering we found 9 number of clusters for Indian news and 10 for American news in case of TF-IDF vectorizer. The above images are for Indian news and American news for some cluster images as well as elbow method graph.

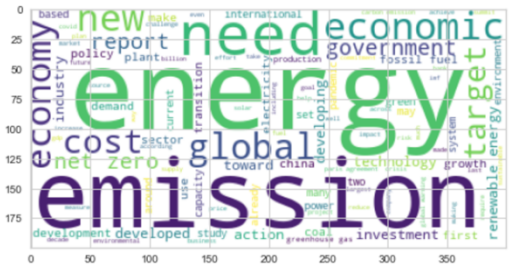
```
visualizer.fit(word_vectors)        # Fit the data to the visual
visualizer.show()        # Finalize and render the figure
```

```
This is for Indian news
```



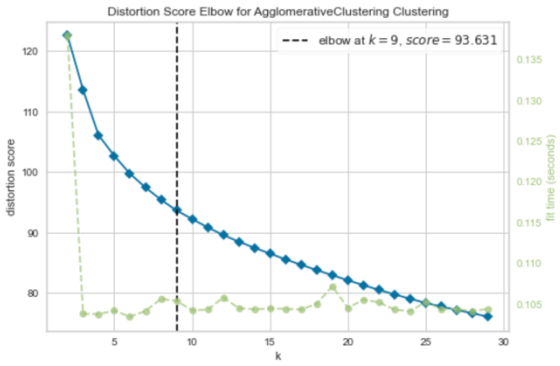

```
wc = WordCloud(stopwords=stopwords,background_color = 'white',m
wc.generate(str_words)
plt.imshow(wc)
plt.show()
```

```
==================================================
Cluster  8  talks about :
```



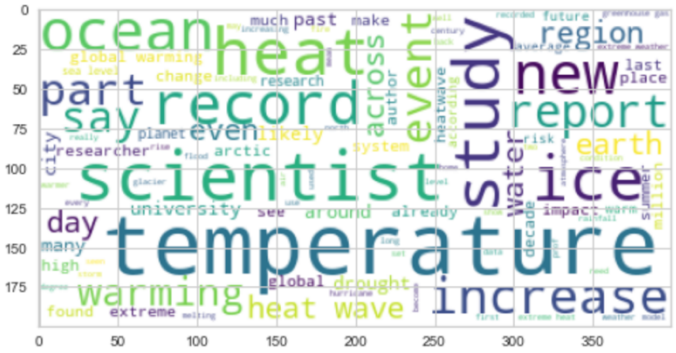

```
visualizer = KElbowVisualizer(model, k=(2,30), timings= True)
visualizer.fit(word_vectors1)        # Fit the data to the visua
visualizer.show()        # Finalize and render the figure
```

```
This is for American news
```




```
wc.generate(str_words)
plt.imshow(wc)
plt.show()
```

```
==================================================
Cluster  9  talks about :
```



Using agglomerative clustering we found 10 number of clusters for Indian news and 9 for American news in case of Word2Vec vectorizer. The above images are for Indian news and American news for some cluster images as well as elbow method graph.

DBSCAN

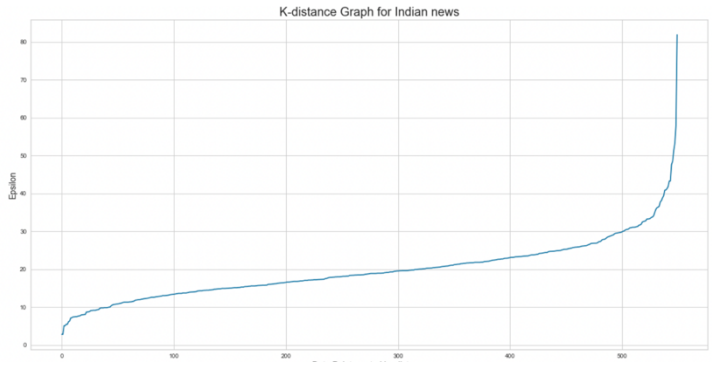
As a general rule, clustering analysis is an Unsupervised learning method that divides the data points into a number of specific groups, such that data points in the same groups have similar properties, and data points in different groups have different properties in some way. The fundamental approach used by all clustering methods is the same, i.e. first, we calculate similarities between data points, and then we use this information to group the data points into groups or batches. The most important feature of DBSCAN is that it is robust to outliers. It also

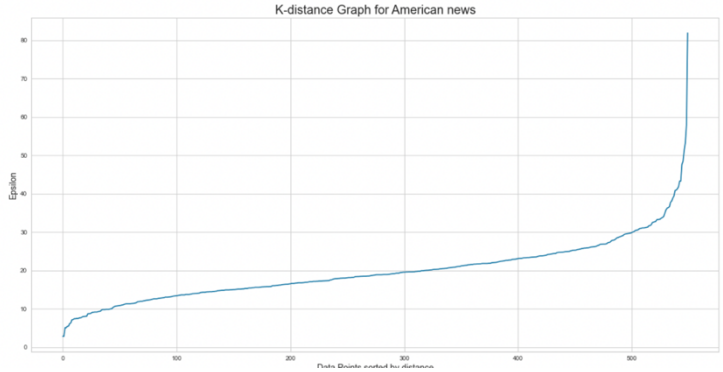
does not require the number of clusters to be told beforehand, unlike K-Means, where we have to specify the number of centroids.

DBSCAN requires only two parameters: the value of epsilon and the number of points to scan. Epsilon is the radius of the circle that will be drawn around each data point in order to check its density, and minPoints is the minimum number of data points that must be contained within that circle in order for that data point to be classified as a Core point. In higher dimensions, the circle is transformed into a hypersphere, with epsilon representing the radius of the hypersphere and minPoints representing the bare minimum number of data points required within the circle.

For example, K-Means (distance between points), Affinity propagation (graph distance), Mean-shift (distance between points), DBSCAN (distance between nearest points), Gaussian mixtures (Mahalanobis distance to centres), Spectral clustering (graph distance), and so on are examples of distance measures.

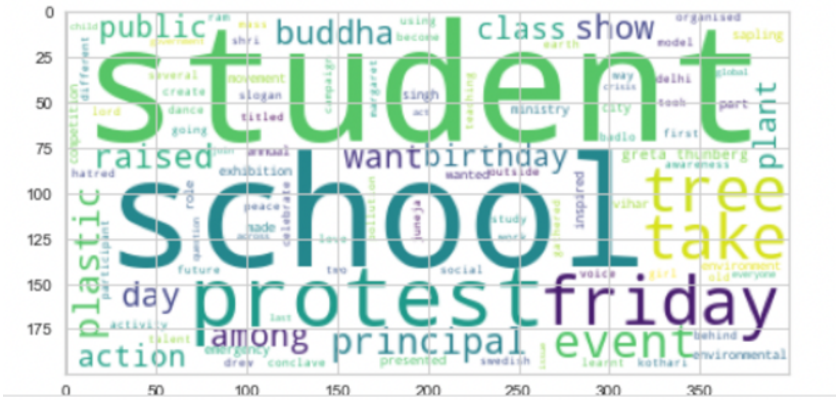
In our approach we first find Nearest neighbors for the 3 vectorized text using values as 550 as it is about the same as the number of rows in Indian news. Then we fit the K nearest neighbor algorithm and sort the distance and indices to plot the graph for epsilon versus the data points sorted by distance for both American and Indian news.





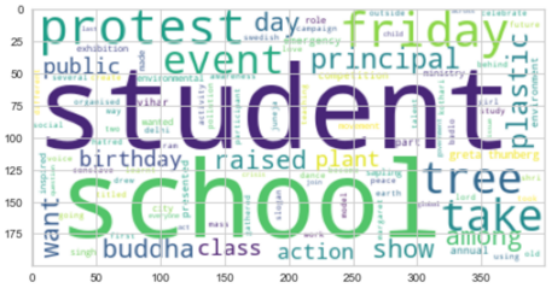
```
    if i== -1:
        continue
    print("="*50)
    print("Cluster ",i+1," talks about :")
    print()

    str_words = ' '.join(df[df["labels"]==i]["text"].values.
    wc = WordCloud(stopwords=stopwords,background_color = 'wh
    wc.generate(str_words)
    plt.imshow(wc)
    plt.show()
```

```
==================================================
Cluster  6  talks about :
```

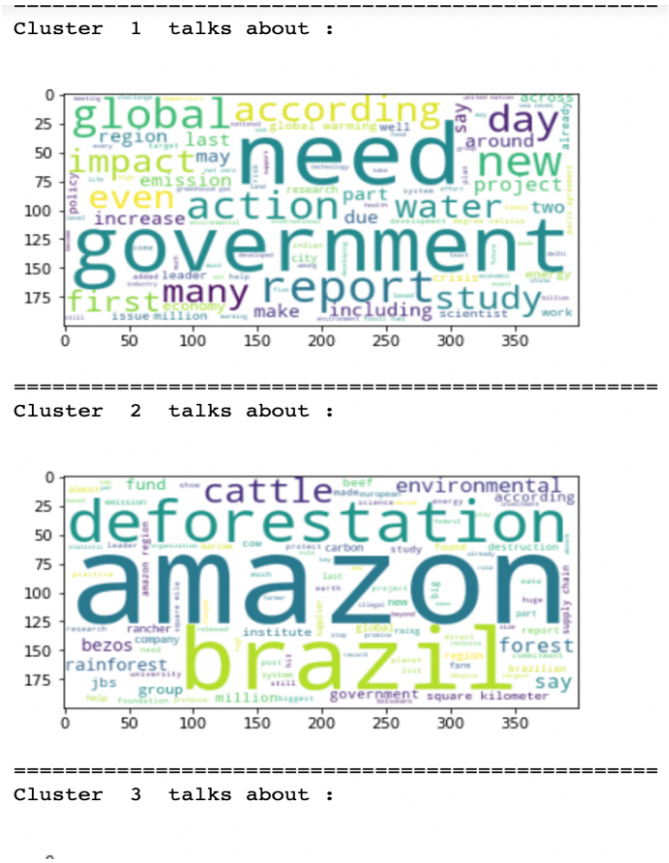


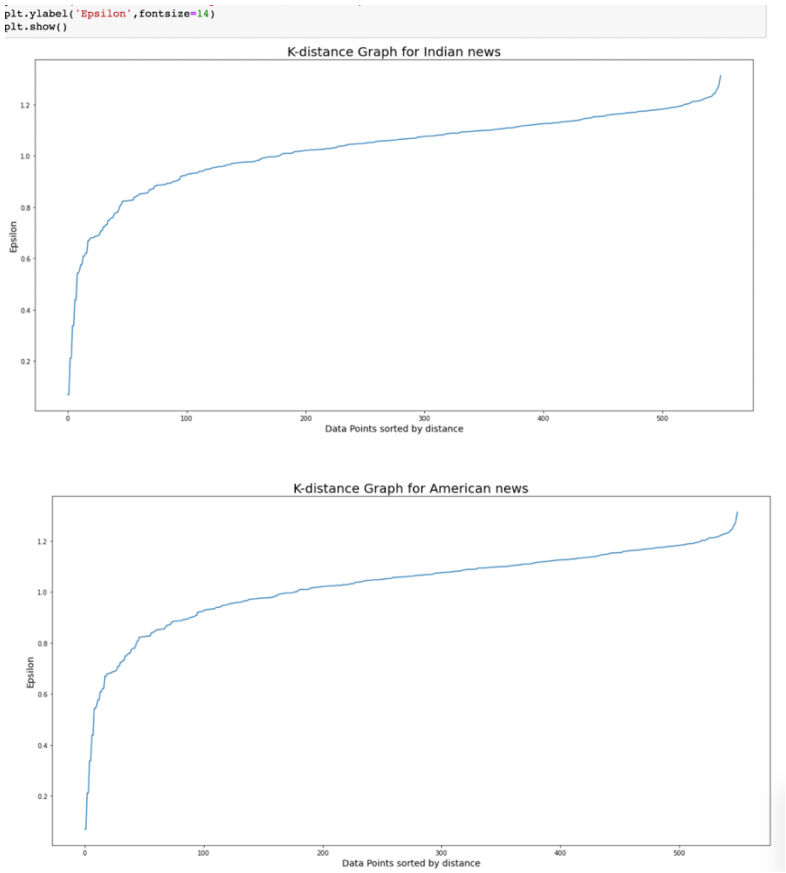

```
f["labels"] = cluster_labels
for i in sorted(list(set(cluster_labels))):

    if i== -1 :
        continue
    print("="*50)
    if i == 6:
        break
    print("Cluster ",i+1," talks about :")
    print()
    str_words = ' '.join(df[df["labels"]==i]["text"].values.tolist())
    wc = WordCloud(stopwords=stopwords,background_color = 'white',max_words = 100)
    wc.generate(str_words)
    plt.imshow(wc)
    plt.show()
```

```
==================================================
Cluster  6  talks about :
```

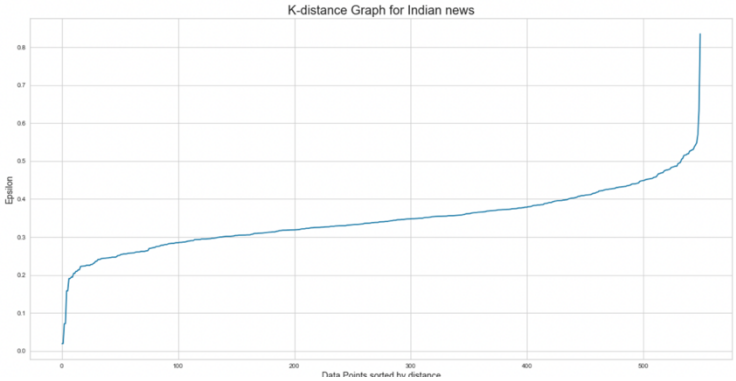


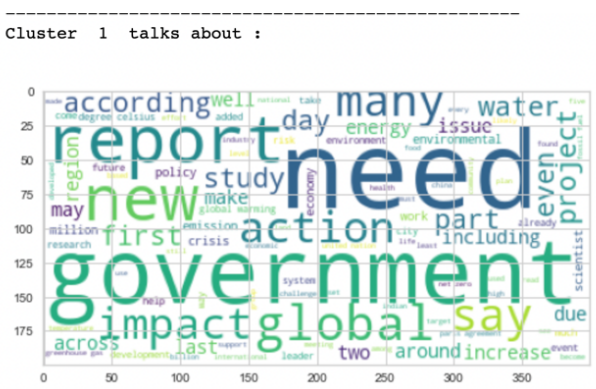
Using DBSCAN clustering we found 6 number of clusters for Indian news and 6 for American news in case of BOW vectorizer. The above images are for Indian news and American news for some cluster images as well as elbow method graph for the epsilon value. We take the epsilon with the maximum curvature of slope.

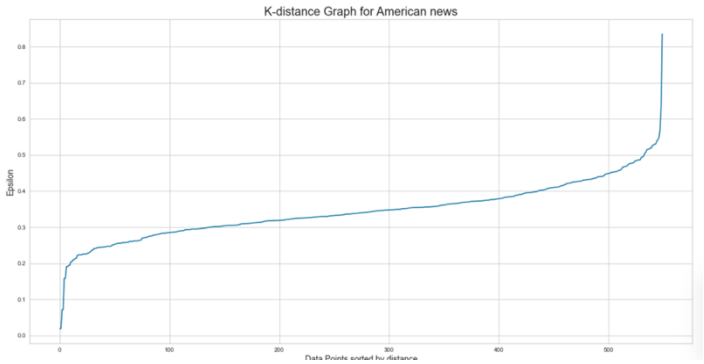


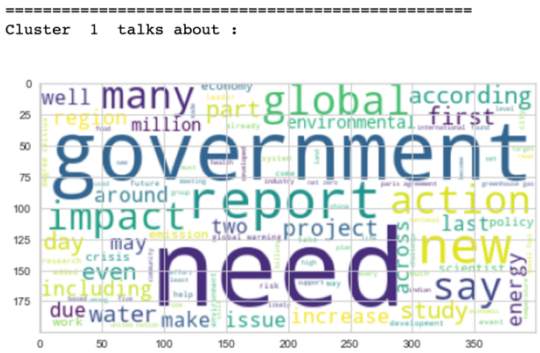


Using DBSCAN clustering we found 5 number of clusters for Indian news and 5 for American news in case of TF-IDF vectorizer. The above images are for Indian news and American news for some cluster images as well as elbow method graph for the epsilon value. We take the epsilon with the maximum curvature of slope.









Using DBSCAN clustering we found 1 number of clusters for Indian news and 1 for American news in case of TF-IDF vectorizer. The above images are for Indian news and American news for some cluster images as well as elbow method graph for the epsilon value. We take the epsilon with the maximum curvature of slope.

We used eps=0.36,eps= 0.7,eps=0.47 for bag of words, TF-IDF and WordVectors respectively in case of Indian news.We used eps=0.3, eps=0.62 ,eps=0.5 for bag of words, TF-IDF and WordVectors respectively in case of American news. We used number of minimum samples as 4 and 3 in some cases based on the past clusters formed and the amount of datapoints being less.

Latent Dirichlet Allocation

Then we move on to LDA (Latent Dirichlet Allocation) for topic modelling of the data. . We found out the best log likelihood and perplexity of the data (both Indian and American news). The lower the log likelihood the better. This was done during grid search CV to tune the

parameters by putting a range of number of topic values, we kept the training to online so if any new data comes for topic modelling , it can make use of the vocabulary of data previously used for newer arriving data. There are two Dirichlet distributions used in the LDA algorithm. In the case of document analysis, there's a distribution over the topics in each document and a distribution over the words in each topic. The LDA topic and word distributions are based on frequency counts of the topics and words in the set of documents being analyzed.

After the gridsearch we take the best models for Indian news and American news having best hyperparameters to fit and transform the Indian news and American news data , thus giving us a topics so we visualize a tabulation with topics as columns and documents as rows. You can see the the tabulation for Indian news and American news respectively below.

	Topic0	Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9	dominant_topic
Doc0	0.010000	0.010000	0.010000	**0.920000**	0.010000	0.010000	0.010000	0.010000	0.010000	0.010000	**3**
Doc1	0.010000	0.010000	0.010000	**0.920000**	0.010000	0.010000	0.010000	0.010000	0.010000	0.010000	**3**
Doc2	0.020000	0.020000	0.020000	**0.830000**	0.020000	0.020000	0.020000	0.020000	0.020000	0.020000	**3**
Doc3	0.020000	0.020000	0.020000	**0.860000**	0.020000	0.020000	0.020000	0.020000	0.020000	0.020000	**3**
Doc4	0.010000	0.010000	0.010000	**0.930000**	0.010000	0.010000	0.010000	0.010000	0.010000	0.010000	**3**

	Topic0	Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9	dominant_topic
Doc0	0.010000	0.010000	0.010000	0.010000	0.010000	**0.910000**	0.010000	0.010000	0.010000	0.010000	**5**
Doc1	0.010000	0.010000	0.010000	0.010000	0.010000	**0.910000**	0.010000	0.010000	0.010000	0.010000	**5**
Doc2	0.010000	0.010000	0.010000	0.010000	0.010000	**0.940000**	0.010000	0.010000	0.010000	0.010000	**5**
Doc3	0.010000	0.010000	0.010000	0.010000	0.010000	**0.920000**	0.010000	0.010000	0.010000	0.010000	**5**
Doc4	0.010000	0.010000	0.010000	0.010000	0.010000	**0.910000**	0.010000	0.010000	0.010000	0.010000	**5**

Then we create a topic to keyword or "term" tabulation for both the datasets. We can see it below.

	ability	able	access	accord	according	account	achieve	achieving	acre	across	...	worth	would	wrote	year	
Topic0	0.101469	0.101869	0.102002	0.102169	0.101792	0.101903	0.101848	0.101888	0.101937	0.101813	...	0.102006	0.101955	0.101484	0.102542	0.1
Topic1	0.102215	0.102162	0.102642	0.103435	0.103111	0.101762	0.102067	0.102089	0.101854	0.103788	...	0.101884	0.103916	0.102490	0.106407	0.1
Topic2	0.101850	0.101998	0.101731	0.102076	0.102720	0.102038	0.102004	0.101822	0.102010	0.102312	...	0.101901	0.102870	0.102200	0.104776	0.1
Topic3	1.857077	2.787661	3.458145	4.652265	9.781367	3.352793	3.878060	2.107006	2.206027	7.769008	...	1.671719	14.483198	2.166072	21.765111	3.5
Topic4	0.101767	0.101956	0.102238	0.102338	0.102525	0.102140	0.102924	0.101997	0.102314	0.102229	...	0.101453	0.103583	0.102177	0.102892	0.1

5 rows × 1000 columns

```
df_topic_keywords = pd.DataFrame(best_lda_model1.components_)
df_topic_keywords.columns = vectorizer1.get_feature_names()
df_topic_keywords.index = topicnames
df_topic_keywords.head()
```

```
/Users/adityagaikwad/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Functi
on get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get
_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

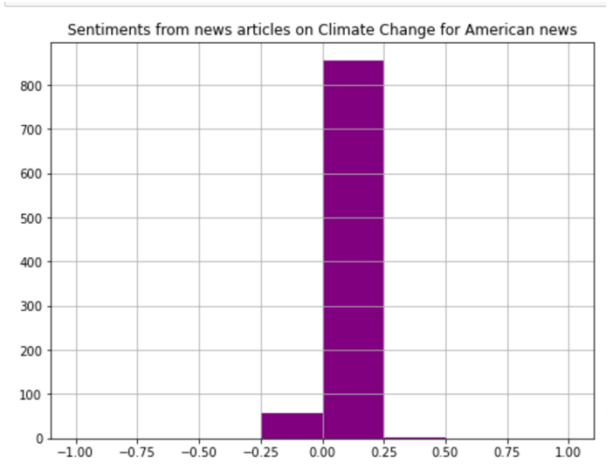
	able	access	according	account	achieve	across	act	action	activist	activity	...	world	worse	worst	would	year	
pic0	0.100084	0.100116	0.100143	0.100113	0.100104	0.100133	0.100095	0.100145	0.100103	0.100094	...	0.100136	0.100091	0.100091	0.100136	0.100141	0
pic1	0.100111	0.100111	0.100192	0.100107	0.100091	0.100094	0.100113	0.100141	0.100135	0.100132	...	0.100231	0.100104	0.100122	0.100205	0.100269	0
pic2	0.100107	0.100094	0.100124	0.100096	0.100101	0.100127	0.100095	0.100118	0.100099	0.100085	...	0.100135	0.100103	0.100095	0.100137	0.100146	0
pic3	0.100143	0.100131	0.100208	0.100082	0.100107	0.100141	0.100173	0.100198	0.100124	0.100124	...	0.100203	0.100112	0.100106	0.100379	0.100305	0
pic4	0.100101	0.100096	0.100130	0.100143	0.100104	0.100145	0.100134	0.100132	0.100096	0.100096	...	0.100157	0.100121	0.100116	0.100141	0.100136	0

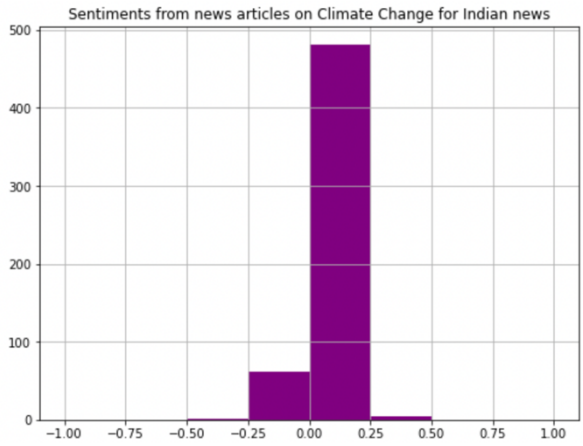
Then we create a proper tabulation of top 15 keywords of every topic based on the topic weights of the terms. You can see the results below, first result being for Indian news and second for American news.

	Important Word 1	Important Word 2	Important Word 3	Important Word 4	Important Word 5	Important Word 6	Important Word 7	Important Word 8	Important Word 9	Important Word 10	Important Word 11	Important Word 12	Impo Wo
Topic 1	thunberg	school	activist	art	europe	target	strike	compared	student	protest	concentration	week	
Topic 2	trump	biden	world	president	cop	conference	thunberg	water	year	debate	country	pandemic	
Topic 3	thunberg	greta	protest	country	action	strike	fire	year	school	read	also	people	
Topic 4	world	year	country	report	global	emission	people	one	also	state	forest	temperature	
Topic 5	forest	project	eia	notification	state	environment	ministry	draft	government	department	minister	area	
Topic 6	sea	ice	specie	year	island	snow	level	record	emission	region	tiger	commission	anta
Topic 7	country	economic	emission	book	report	environment	disease	government	landslide	zero	new	need	
Topic 8	ice	amazon	region	sea	antarctica	imf	project	study	level	deforestation	warming	square	implemen
Topic 9	forest	crore	oil	farmer	bird	also	wetland	state	district	coral	threat	year	
Topic 10	biden	mumbai	trump	california	president	rain	state	day	recorded	weather	fire	wildfire	temper

	Important Word 1	Important Word 2	Important Word 3	Important Word 4	Important Word 5	Important Word 6	Important Word 7	Important Word 8	Important Word 9	Important Word 10	Important Word 11	Important Word 12	Important Word 13
Topic 1	drought	lake	que	water	los	conference	county	la	california	heat	council	china	space
Topic 2	degree	country	carbon	emission	heat	global	gas	year	goal	water	methane	celsius	advertisement
Topic 3	fashion	que	fossil	fuel	emission	company	morning	warming	celsius	la	hurricane	degree	goal
Topic 4	nuclear	would	state	advertisement	fire	new	program	biden	year	flood	federal	united	community
Topic 5	report	nbc	fire	morning	heat	state	community	could	extreme	farmer	china	released	health
Topic 6	emission	government	carbon	world	say	country	year	people	energy	would	new	cop	gas
Topic 7	insulate	britain	protester	police	sus	protest	road	traffic	court	activist	highway	london	officer
Topic 8	vehicle	indigenous	system	amazon	federal	supplier	leather	america	land	cattle	crop	quality	state
Topic 9	ice	percent	heat	year	world	city	johnson	study	people	new	get	island	council
Topic 10	que	los	la	para	una	por	del	dijo	con	como	est	sus	advertisement

We have done polarity sentiment visualisation of both the datasets using Textblob . Results of which can be clearly understood by the images below. The polarities being from –1 to 1 . -1 suggesting negative sentiment and 1 suggesting positive sentiment.





Analysis Of Experiment Results And Conclusion

We get 8 , 9 and 4 clusters for bag of words, TF-IDF and Word2Vec respectively. So we decide to take an approximate average of 8 clusters for the Kmeans algorithm for Indian news. For American news the same case of Kmeans with different vectorizers is giving results as 8 , 6 and 4 for bag of words, TF-IDF and Word2Vec respectively. So we take an approximation of 6 clusters for American news by the mean of these values.

In case of GMM with different vectorizers for the Indian news.SO we get 9 , 9 and 9 clusters for bag of words, TF-IDF and Word2Vec respectively.So we decide to take an approximate average of 9 clusters for the GMM algorithm for Indian news. For American news the same case of GMM with different vectorizers is giving results as 7 , 9 and 4 for bag of words, TF-IDF and Word2Vec respectively. So we take an approximation of 7 clusters for American news by the mean of these values.

In Agglomerative clustering with different vectorizers for the Indian news we get 10 , 9 and 9 clusters for bag of words, TF-IDF and Word2Vec respectively. So we decide to take an approximate average of 9 clusters for the Agglomerative algorithm for Indian news. For American news the same case of Agglomerative clustering with different vectorizers is giving results as 10 , 10 and 9 for bag of words, TF-IDF and WordVectors respectively.So we take an approximation of 10 clusters for American news by the mean of these values.

In case of DBSCAN we get 6 and 6 clusters for Indian and American news respectively in case of BOW vectorizer. We get 5 and 5 clusters for Indian and American news respectively in case of TF-IDF vectorizer. We get 1 cluster for both Indian and American news in case of Word2Vec vectorizer.We do external evaluation of all the clusters and find out that all the 3 vectorizers perform well with GMM and K means.

The Word2Vec seems to give lesser number of unique clusters compared to other vectorizers but other two vectorizers have been performing well in the external evaluation as we look at the cluster word clouds and the uniqueness of the content. Word2Vec seems to have worked arbitarily in the DBSCAN method giving just one cluster but the other two vectorizers have performed comparatively well.

If we used the pre-trained word2vec embedding and the text corpus is not very specialized, the semantic relationships of words are maintained. Therefore, word2vec is definitely useful for clustering. Why? Because the geometry of data in the new vector space holds important information that is very crucial for clustering purposes. No. If you trained the word2vec embedding on your own small-size text corpus or your text corpus which is definitely a case in our work and has a very detailed focus (e.g., Climate change data from different parts of news), I highly recommend to not use any clustering technique with Word2Vec or use them very cautiously. Because there is no guarantee that the geometry of data remains untouched in the new vector space.

Internal Evaluation:-

To evaluate the internal metrics and do a final definite analysis we take the average clusters for all the algorithms that we found above for both datasets and use those values as an input as well as reference in the case of DBSCAN algorithm to find the Silhouette Score, Calinski Harabasz Score, and Davies Bouldin Score.

The Silhouette Coefficient is calculated for each sample and is made up of two scores (shown below), with a higher score indicating a model with more distinct clusters.

1)The average distance between a sample and the rest of the class's points. This number indicates how near points in the same cluster are.

2)The average distance between a sample and the nearby cluster's other points. The distance between points in various clusters is measured by this score.

The Silhouette Coefficient *s* for a single sample is then given as:

$$s = \frac{b - a}{max(a, b)}$$

The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample.

The score ranges from -1 to +1 for faulty clustering to extremely dense clustering. Overlapping clusters are indicated by scores around 0. When clusters are dense and well spaced, the score is greater, which corresponds to a standard definition of a cluster.Convex clusters have a greater

Silhouette Coefficient than other types of clusters, such as density-based clusters like those acquired by DBSCAN.

The Calinski-Harabasz index, also known as the Variance Ratio Criterion, is the ratio of the total of within-cluster and inter-cluster dispersion for all clusters; the higher the score, the better the results for clustering.

For a set of data E of size n_E which has been clustered into k clusters, the Calinski-Harabasz score s is defined as the ratio of the between-clusters dispersion mean and the within-cluster dispersion:

$$s = \frac{\mathrm{tr}(B_k)}{\mathrm{tr}(W_k)} \times \frac{n_E - k}{k - 1}$$

where $\mathrm{tr}(B_k)$ is trace of the between group dispersion matrix and $\mathrm{tr}(W_k)$ is the trace of the within-cluster dispersion matrix defined by:

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T$$

with C_q the set of points in cluster q, c_q the center of cluster q, c_E the center of E, and n_q the number of points in cluster q.

The Davies Bouldin Score represents the average 'similarity' of clusters, where similarity is a metric that relates cluster distance to cluster size. A model with a lower Davies-Bouldin index has a better separation between the clusters.

The index is defined as the average similarity between each cluster C_i for $i = 1, \ldots, k$ and its most similar one C_j. In the context of this index, similarity is defined as a measure R_{ij} that trades off:

- s_i, the average distance between each point of cluster i and the centroid of that cluster – also know as cluster diameter.
- d_{ij}, the distance between cluster centroids i and j.

A simple choice to construct R_ij so that it is nonnegative and symmetric is:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

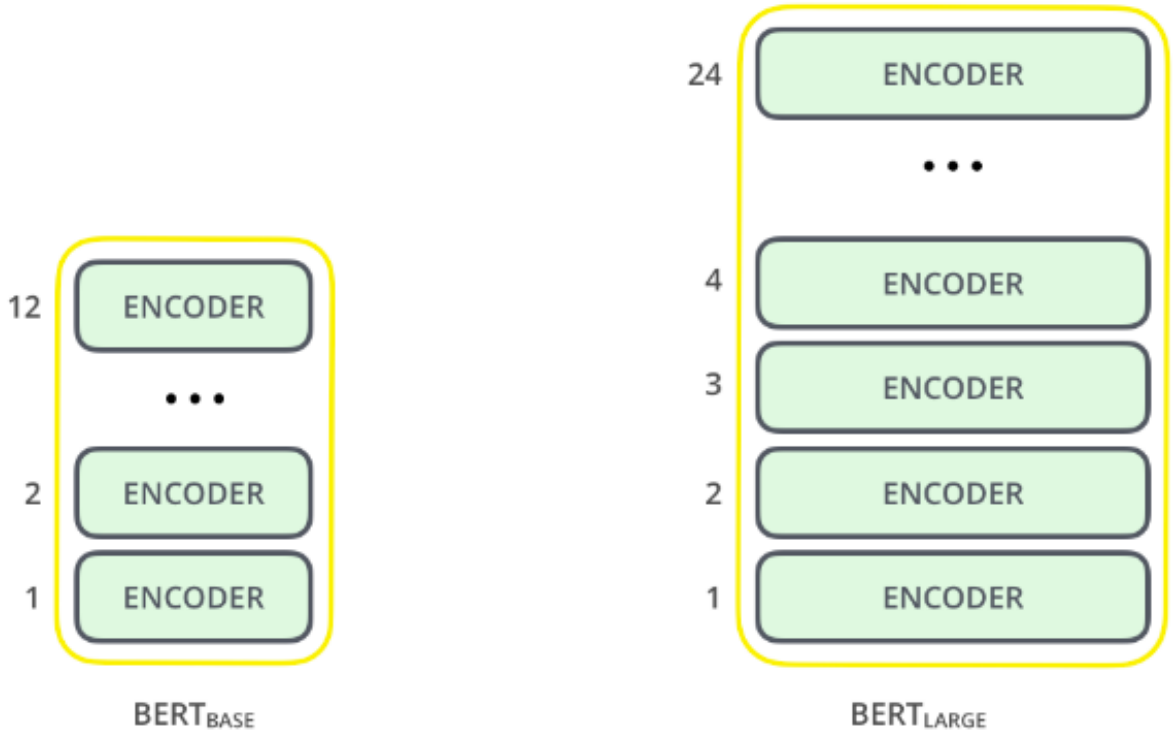
Then the Davies-Bouldin index is defined as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}$$

BERT

"BERT stands for Transformers' Bidirectional Encoder Representations." It is intended to condition both left and right context to pre-train deep bidirectional representations from unlabeled text. As a result, with just one additional output layer, the pre-trained BERT model may be fine-tuned to generate models for many applications.

So due to BERTs fame in the NLP industry and its advantage we have used it as our transformer for dataset to produce BERT encodings. We have also done this step so that for internal evaluation we get the a common ground by using a common transformer for all datasets to be run on all clustering models. This is also a good approach to get a final clustered product industry-wise , so we tried to make the results and final clustering crisper ,so we can create a



product for future use. Below is the picture of BERT’s architecture.

We calculate the above mentioned three metrics for Kmeans with both datasets using 8 clusters for Indian news and 6 clusters for American news. These are averages of the number of clusters we found out before with the three vectorizers during external evaluation. We get the following scores for Indian news and American news respectively as we can see in the figures below.

```
score_kemans_d = davies_bouldin_score(bert_encodings, predictions)
print('Silhouette Score: %.4f' % score_kemans_s)
print('Calinski Harabasz Score: %.4f' % score_kemans_c)
print('Davies Bouldin Score: %.4f' % score_kemans_d)
print("The above scores are for Indian news using Kmeans and optimum cluster")
```

```
Silhouette Score: 0.0638
Calinski Harabasz Score: 23.6657
Davies Bouldin Score: 3.0689
The above scores are for Indian news using Kmeans and optimum cluster
```

```
#Calculation of scores for American news using Kmeans
kmeans_2 = KMeans(n_clusters=6,random_state= 10)
# Use fit_predict to cluster the dataset
predictions = kmeans_2.fit_predict(bert_encodings1)
# Calculate cluster validation metrics
score_kemans_s = silhouette_score(bert_encodings1, kmeans_2.labels_, metric='euclidean
score_kemans_c = calinski_harabasz_score(bert_encodings1, kmeans_2.labels_)
score_kemans_d = davies_bouldin_score(bert_encodings1, predictions)
print('Silhouette Score: %.4f' % score_kemans_s)
print('Calinski Harabasz Score: %.4f' % score_kemans_c)
print('Davies Bouldin Score: %.4f' % score_kemans_d)
print("The above scores are for American news using Kmeans and optimum cluster")
```

```
Silhouette Score: 0.0534
Calinski Harabasz Score: 45.9321
Davies Bouldin Score: 3.0311
The above scores are for American news using Kmeans and optimum cluster
```

```
Silhouette Score: 0.0554
Calinski Harabasz Score: 21.6489
Davies Bouldin Score: 3.1182
```

```
# gaussian mixture clustering for American news
from numpy import unique
from numpy import where
from sklearn.mixture import GaussianMixture
from matplotlib import pyplot
# define the model
model = GaussianMixture(n_components= 7,covariance_type= "full", ran
# fit the model
model.fit(bert_encodings1)
# assign a cluster to each example
yhat = model.predict(bert_encodings1)
# retrieve unique clusters
clusters = unique(yhat)
# Calculate cluster validation score
score_dbsacn_s = silhouette_score(bert_encodings1, yhat, metric='euc
score_dbsacn_c = calinski_harabasz_score(bert_encodings1, yhat)
score_dbsacn_d = davies_bouldin_score(bert_encodings1, yhat)
print('Silhouette Score: %.4f' % score_dbsacn_s)
print('Calinski Harabasz Score: %.4f' % score_dbsacn_c)
print('Davies Bouldin Score: %.4f' % score_dbsacn_d)
```

```
Silhouette Score: 0.0538
Calinski Harabasz Score: 39.9263
Davies Bouldin Score: 3.0221
```

We calculate the metrics again for GMM with both dataset using 9 clusters for Indian news and 7 clusters for American news. These are averages of the number of clusters we found out before with the three vectorizers during external evaluation. We get the following scores when we use it with Bert encodings. The results are of Indian news followed by American news for Bert encodings.

```
print('Davies Bouldin Score: %.4f' % score_AGclustering_d)
```

```
Silhouette Score: 0.0563
Calinski Harabasz Score: 19.9324
Davies Bouldin Score: 3.1548
```

We calculate the metrics again for Agglomerative clustering with both datasets using 9 clusters for Indian news and 9 clusters for American news. These are averages of the number of clusters we found out before with the three vectorizers during external evaluation. We get the following scores when we them with Bert encodings. The results ae of Indian news followed by American news for Bert encodings.

```
# Agglomerative clustering for American news
from numpy import unique
from numpy import where
from sklearn.cluster import AgglomerativeClustering
from matplotlib import pyplot
# define the model
model = AgglomerativeClustering(n_clusters=9)
# fit model and predict clusters
yhat = model.fit(bert_encodings1)
yhat_2 = model.fit_predict(bert_encodings1)
# retrieve unique clusters
clusters = unique(yhat)
# Calculate cluster validation metrics
score_AGclustering_s = silhouette_score(bert_encodings1, yhat.l
score_AGclustering_c = calinski_harabasz_score(bert_encodings1,
score_AGclustering_d = davies_bouldin_score(bert_encodings1, yh
print('Silhouette Score: %.4f' % score_AGclustering_s)
print('Calinski Harabasz Score: %.4f' % score_AGclustering_c)
print('Davies Bouldin Score: %.4f' % score_AGclustering_d)
```

```
Silhouette Score: 0.0359
Calinski Harabasz Score: 32.0871
Davies Bouldin Score: 3.0444
```

In case of DBSCAN we got an average EPS value of around 0.5 from the average of all the EPS that we found before for the three vectorizers using the elbow method. Here we repeat the elbow method to find the epsilon value for bert encodings of both datasets and find out that we get an approximate eps of 0.42 for Indian news and 0.43 for American news. We get the following scores of Indian news followed by American news given below.

```
print( Davies Bouldin Score: %.4f  % score_dbsacn_d)
```

```
Silhouette Score: 0.0248
Calinski Harabasz Score: 3.8568
Davies Bouldin Score: 1.3265
```

```
# dbscan clustering for amrican news
from numpy import unique
from numpy import where
from sklearn.cluster import DBSCAN
from matplotlib import pyplot
# define dataset
# define the model
model = DBSCAN(eps=0.43, min_samples= 3)
# rule of thumb for min_samples: 2*len(cluster_df.columns)
# fit model and predict clusters
yhat = model.fit_predict(bert_encodings1)
# retrieve unique clusters
clusters = unique(yhat)
# Calculate cluster validation metrics
score_dbsacn_s = silhouette_score(bert_encodings1, yhat, met
score_dbsacn_c = calinski_harabasz_score(bert_encodings1, yh
score_dbsacn_d = davies_bouldin_score(bert_encodings1, yhat)
print('Silhouette Score: %.4f' % score_dbsacn_s)
print('Calinski Harabasz Score: %.4f' % score_dbsacn_c)
print('Davies Bouldin Score: %.4f' % score_dbsacn_d)
```

```
Silhouette Score: -0.1445
Calinski Harabasz Score: 6.7541
Davies Bouldin Score: 2.0120
```

The highest Silhoutte score is produced by Kmeans algorithm for Indian news and highest Calinski Harbasz score is also achieved by Kmeans algorithm.The lowest Davies Bouldin score is achieved by DBSCAN algorithm. This suggests DBSCAN had the most unique clusters in case of Indian news. So according to our analysis K means is a clear winner for the Indian news

,but an external evaluation might still be needed to make sure we get the perfect clusters and the shortcomings of uniqueness of cluster which DBSCAN fulfills can also be evaluated by externally having a look at the clusters.

The highest Silhoutte score is produced by GMM algorithm for American news but it's a difference of just 0.0002 .Its almost equal to Kmeans's silhoutte score .The highest Calinski Harbasz score is achieved by Kmeans algorithm.The lowest Davies Bouldin score is achieved by DBSCAN algorithm. This suggests DBSCAN had the most unique clusters in case of American news too. So according to our analysis K means is a winner for the American news ,but an external evaluation might still be needed to make sure we get the perfect clusters and the shortcomings of the uniqueness of the cluster which DBSCAN fulfills can also be evaluated by externally having a look at the clusters. Actually,DBSCAN is fundamentally based on density of cluster so it is naturally going to giving lowest Davies Bouldin score due to its convex nature.

Also the general sentiment analysis of text articles suggests that Indian news has more negative sentiments compared to America. We can conclude that various countries have different concerns with the climate change. We can observe that developed nations like the US are concerned more with the carbon emissions, fire etc. We can notice that tropical counties like India has monsoons, air pollution,forest ,Ministry, government, ice levels due to global warming as concerns.

What part of your methodology worked (or didn't work)?

So basically the two vectorizers Bag Of Word vectorizers and TF-IDF vectorizers have worked properly but there seemed to be a problem with Word2Vec for clustering text. The methodology used in the end worked pretty well on getting the right algorithm for clustering news articles of climate change.

Why did your methodology work or (didn't work)?

Our methodology worked as we used a common encoding technique(BERT) that's pretty intensive and advanced for vectorizing the text and then using different algorithms with an average number of clusters from different vectorizers which we used during external manual evaluation before .These three(BOW,TF-IDF,Word2Vec) vectorizers are used as each vectorizer has its own pros and cons. So using multiple of them really describes how the data

reacts to various parameters of vectorization which helps in analysis in the end. So these average number of clusters(average of clusters found for one clustering technique for the three vectorizers) act as a cascading filter in our algorithm to move to our next step of again using those cluster numbers as a reference for calculating internal metrics. This thus creates a crisp pipeline that goes through a series of tests and observations before reaching towards any final result .After this filtering as we find out the three internal evaluation metrics we get accurate results. Another reason for finding better results is that Indian news has less dense and number of data points compared to American news which has a stacking of various American data increasing the number of data points as well as variability. This gives us a highly variable data for our methodology thus proving that our results are producing after a proper filtration with variable conditions.

How to improve?

We can improve the results more by doing an external evaluation again of all the results of the newly found best algorithms. We can also use these newly found clusters to manually label the data and do a sentiment analysis of all the data. We can do a general as well as an indepth emotional sentiment analyis and create a sentiment predicting model ,so we can use this in the industry.

How to utilize your results? What business insights can be derived from your analysis?

So we can create a complete product which clusters , analyses and does sentiment analysis of different countries. This could be a very beneficial product for the market. This could be marketed to different websites which show analysis of climate change data of countries. This can be also used by certain government organisations and lastly by news channels for showing an analytical approach of sentiments and news of different countries

References

- Samuels A., Mcgonical J., “News Sentiment Analysis”
- **https://neptune.ai/blog/sentiment-analysis-python-textblob-vs-vader-vs-flair**
- **https://emergentalliance.org/sentiment-analysis-of-newspapers/**
- **https://www.kdnuggets.com/2018/08/emotion-sentiment-analysis-practitioners-guide-nlp-5.html**