American Presidents' Personality Analysis and Ranking prediction through Semantic Speech Analysis

Problem Statement

The objective is to employ semantic analysis techniques on historical speeches delivered by American Presidents to group them into clusters based on the content and underlying themes. Additionally, the goal is to create a predictive model that can determine the cluster affiliation of a new presidential candidate based on their speech content, leveraging patterns identified from previous presidential speeches.

Furthermore, the project aims to develop a ranking prediction model that estimates the potential rank or position of a new presidential candidate within the established clusters. This prediction will be based on the rankings or classifications derived from the speeches of the existing American Presidents.

Project Objectives

- To group the American Presidents into clusters based on the semantic analysis of the speeches given by them
- If a new presidential candidate gives some speech we should be able to predict the cluster to which he belongs to based on the previous provided speeches
- To predict the ranking of the new presidential candidate based on the rankings of the provided presidents.

Methology

Data collecting

- Data collected by scraping the official website of University of Virginia (Miller Center) which contains the records of speeches given by American Presidents.
  Link : https://www.presidency.ucsb.edu/documents/presidential-documents-archive-guidebook/presidential-campaigns-debates-and-endorsements-3
- Data was also collected from the datasets avaliable at Kaggle.

Merging of Data and Data Exploration

- Data from these different sources were not compatible to merge with each other.
- Merged them by making appropriate conversions.
- The final dataset was then explored to see the amount of data collected and checking whether the data is representative. (Meaning all the American presidents are covered in the data or not.).

Data Preprocessing

Data Cleaning : Removal of punctuation marks , removal of ASCII characters , removal of urls , etc. Making all the chracters lowercase.

Tokenizing the Data : Breaking it down into smaller units, such as words or phrases, to analyze or process it more effectively.

Removing the stop words and other unnecessary words

Stemming the Data : Reducing words to their root form, aiding in text analysis by grouping variations of a word together.

Converting the tokens into string and unidecoding it

Vectorizing the Data : Converting text data into numerical vectors based on term frequency-inverse document frequency, representing each term's importance within a speech corpus.

Finally the data preprocessing part of the data is completed and we wil now move on to the data modelling part.

Unsupervised Data Clustering

We will use K-means clustering to cluster our data into small groups.

Why to use this ???

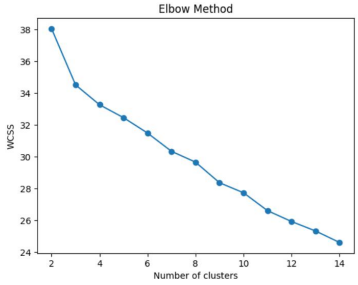
> **No prior assumptions:** K-means doesn't require prior knowledge about the dataset, making it suitable for unsupervised learning.
> **Automatic pattern detection:** It automatically identifies patterns in data by iteratively assigning data points to clusters and optimizing centroids.
> **Fast convergence:** Typically converges quickly, providing results even with high-dimensional data.

But still there is an issue ??

We do not know that how many clusters of data are there in our data. How to know that ?

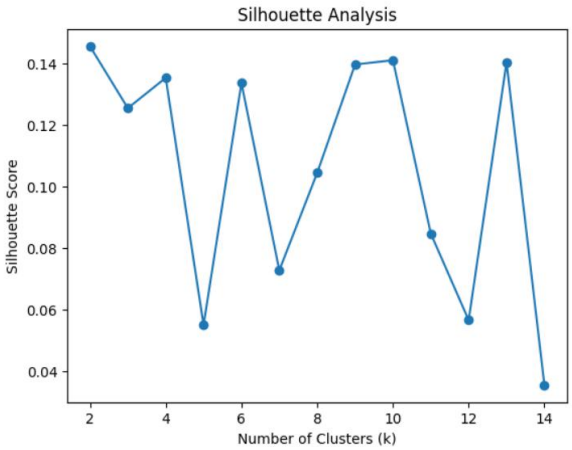
Elbow Method

The elbow method in K-means involves plotting the number of clusters (k) against the sum of squared distances (SSD) of points to their centroids. It aims to find the k value where the decrease in SSD sharply changes or forms an "elbow," representing the optimal number of clusters for effective grouping without overfitting or underfitting the data. This point indicates the most significant improvement in clustering performance while balancing complexity.



Sihouette Score

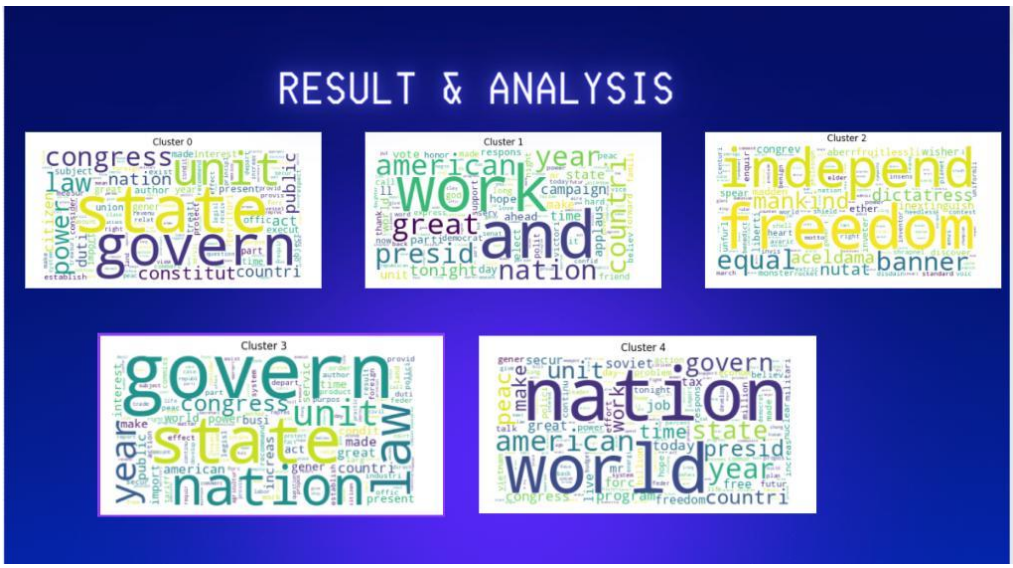
The silhouette score measures the cohesion and separation of clusters based on distances between data points.Higher silhouette scores (ranging from -1 to 1) indicate better-defined clusters.To find the appropriate k (number of clusters), iterate through different k values and select the one with the highest average silhouette score.An ideal k yields a silhouette score close to 1, indicating well-separated and distinct clusters.

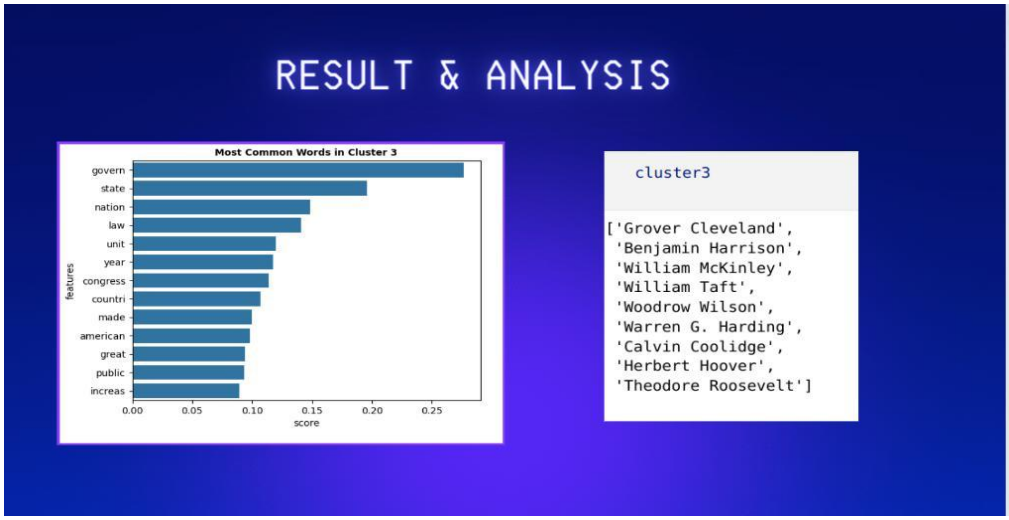


From this we found out that our data is best clutered into five groups.

Results and Analysis

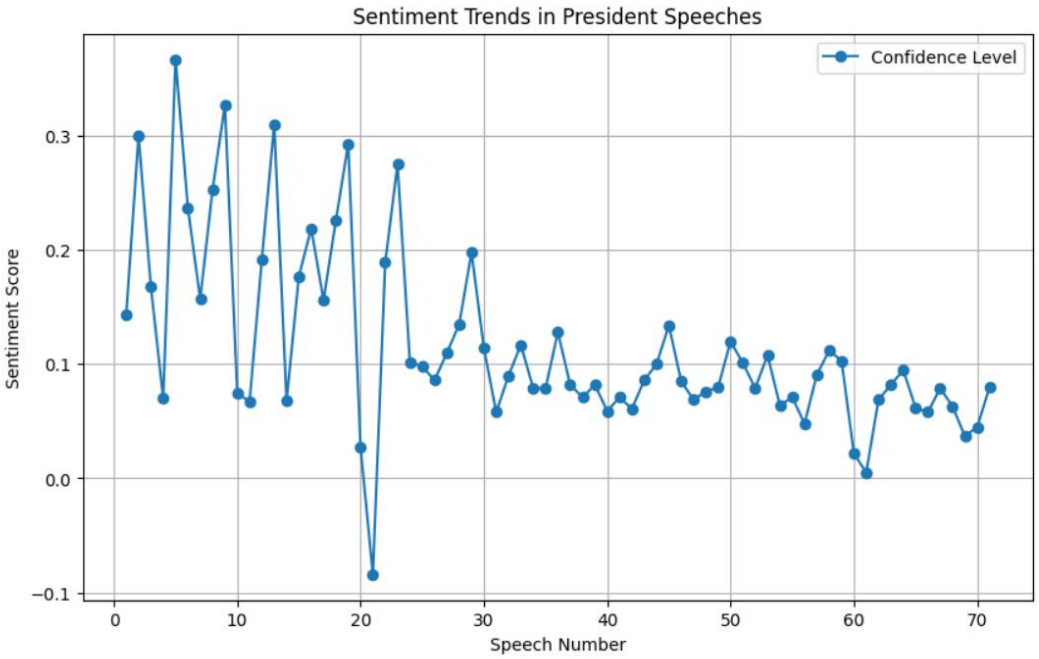
We succesfully divided the American presidents into five groups according to their presonality traits in their speeches.





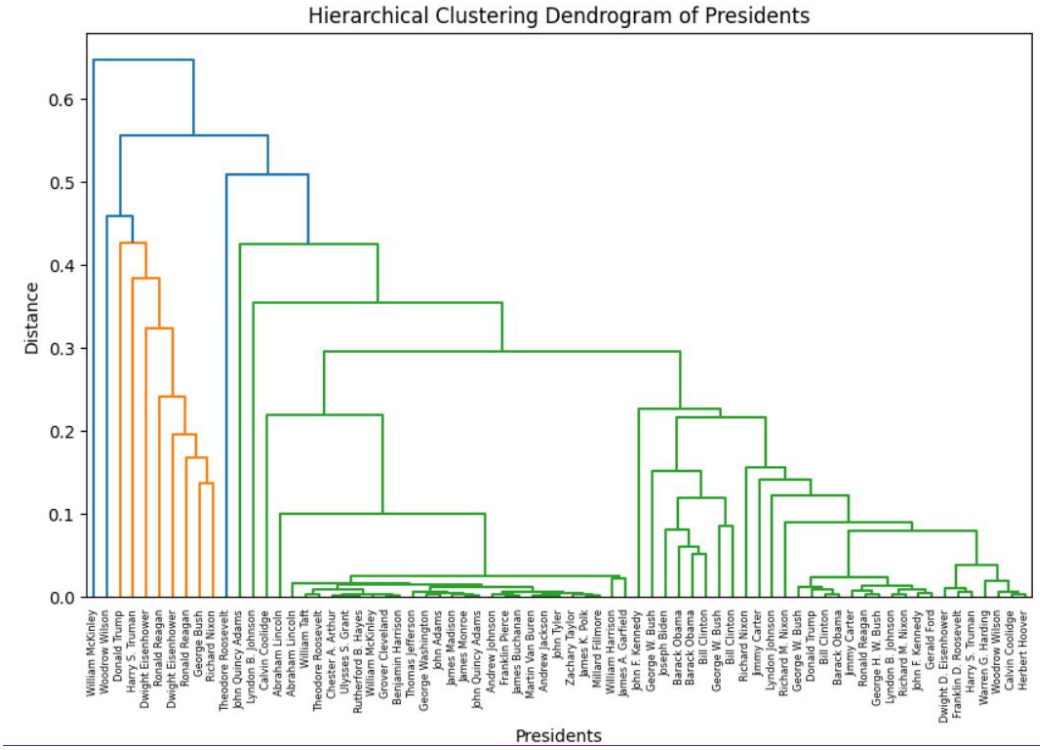
Now we will do analysis about the confidence level in the speeches of the presidents using the **SentimentIntensityAnalyzer** function of **nltk** library.

From which we got the following results:



We can see the confindence level of president in each of the speeches.

Dendogram Analysis



Final Application

Now we manually collected the data of ranking of the American President from the website.
Link : https://www.c-span.org/presidentsurvey2021/?page=overall

And created a csv file named **scores.csv**

Now if any other new speech comes to us we can detect that it belongs to which cluster from one of this five clusters and this will give us insights about the new candidate. Moreover we can also find the ranking of the candidate using the top 5 nearest neighbours.

```
# Speech by George Washington
speech = "Whereas it appears that a state of war exists between Austria, Prus
tell_cluster_and_popscore(speech)
```

```
Speech shows resemblance to Cluster  0
Predicted Popularity Score of this speech :  567.2
```

It detected correct cluster as George Washington belonged to the first cluster

Summary

The project aimed to categorize American Presidents into clusters based on semantic analysis of their speeches, utilizing techniques such as data scraping, merging, cleaning, and vectorizing to prepare the dataset. Employing K-means clustering with an optimal cluster count of five, the Presidents were effectively grouped based on speech content. Furthermore, sentiment analysis gauged confidence levels within these speeches.A crucial component was gathering and classifying Presidents' ranks in order to forecast the ranking of a new contender by figuring out which cluster they belonged to and how close they were to the five closest neighbours in that cluster. Using previous speeches as a guide, this technique provides insights about a prospective candidate's perceived personality attributes and possible rating. With the help of the constructed model, incoming speeches may be quickly analysed and predicted, providing insightful information about a candidate's location within pre-existing clusters and possible rating in relation to previous Presidents.

Explanation of the code

So till now we have the bird view of my project but now I would like to explain my project by explaining the each cell of my Jupiter notebook

In the first cell of my Jupyter notebook I am installing the required libraries in my Conda environment so that I can use them for preprocessing and analysing my data

In the second cell of my Jupyter notebook I am importing the dependency from this libraries

The first library which I have imported is Pandas library. The use of Pandas library is to manipulate and convert the data in such into a data frame process on the data it is also helpful to see some of the basic information about the data.

The second library which I have imported is beautiful soup library so when we have to scrap some of the data from website we use this library because this library brings the data in a tree format so that we can extract the relevant data using the tag of the data since I am doing the website scraping in this project and I will be scraping some of the president speeches from the official website that's why I am using beautiful soup library

The third library which I am importing is NTLK so when so ever we work on some human language data will use this library which is natural language tool kit which have most of the features implement it inside it to work on human language data since we are working on the speeches of American president which are in English and are human language data that is why we are using ntlk library.

The fourth library which I am importing is unidecode library so as we know that in English sometimes some letters have some cap on their top or are belonging from some different language such as French or some other words so to convert those letters into English we require this unique decode a library the main purpose of this library is to convert every character to the letter in English

The fifth library which I am importing is re. The use of this library is to find some expression in a Corpus since we are going to remove things then we have to search for https so this will be done by this library

The six library which I am importing is TFIDF vectorizer since a machine learning model cannot understand human language that is why we have to process that data into numerical data so that are machine learning model or any model can work on it so for converting a human data into numerical data we are going to use this TFT vector I which is going to convert our data into numerical data.

Data collection part

Now I am going to explain how I collected data for my project. Project I required the speeches of the American Presidents. One of the sources of these features was the website. The link of the website from which I collected the speeches is https://www.presidency.ucsb.edu/documents/presidential-documents-archive-guidebook/presidential-campaigns-debates-and-endorsements-3 .

This website itself does not contains does speeches but it contain the link of the pages in which the features are present so I first of all collected the links of other website in which the actual features are present. Then I went to that website and then I scraped the speeches.

This website contains only the speeches from 1900 to 2022 but the also required the speeches from the past presidents of America that is why we are using another resource to get speeches. We are collecting the remaining speeches from Kaggle data set. Since we have collected inform the speeches from two different sources they are not compatible with each other to get merged. In order to merge them first of all I create a dataframe from the scraped speeches. Then I removed the additional information from the Kaggle dataset and made them compatible to each other by renaming their columns. Then I merged the data to form the final data.

Data Exploration Part

We will check that whether the collected data is representative or not. I checked the number of speeches it contains and the number of American Presidents whose speeches are there. I assured that most of the American Presidents are covered. Because if we cover only a handful of Presidents the data will not be representative and it lead to formation of a bias clustering model.

Data Processing Part

Data Cleaning

In this stage we will remove all the ASCII characters form our speeches then we will remove all the commas from our speeches. After removing commas we will remove the new line characters from our speeches. After removing these unnecessary characters from our speech we will convert each letter of our speech to lower case.

Now we going to use the **re** library.

Use of ‘re’ library

re stands for regular expression. This library is used to find a particular pattern in our string or corpus. This library detects that particular pattern and is able to substitute that pattern with any other string you want.

We will use this library to find expression to find links in our speeches by finding the patterns: http and www and after finding this patterns we are going to remove the links from our speeches.

Similarly using this library we are going to remove the special characters and remove numbers.

We will also remove words concatenated with numbers using the same library. We are also going to remove email ids and mentions if present.

After doing all of this the Data cleaning part is completed.

DATA TOKENIZING

After cleaning the data we are going to tokenize the data. What does tokenization means ? Tokenization means to break our corpus into small tokens or words or phrases. We will break our speeches into small tokens. Why tokenization is done ? Tokenization is done because now all words in the corpus are important. We are going to remove the unimportant tokens and processing the tokens one by one is more effective for machine rather than from looking the whole corpus at once.

REMOVING THE STOP WORDS AND OTHER UNNECESSARY WORDS

After converting the corpus into list of tokens we are now going to remove the stop words from the corpus. What are stopwords ? Stopwords are words which are very commonly used in a language. These words does not carry any important meaning or context on their own but they are only act as a connector for the other words. So we are going to keep this words in our token. The model will consider that this are important words because they are getting repeated again and again. But this is not the case. That is why to avoid this confusion we remove this words as by removing this words the context of the corpus is not changed much and the earlier mentioned problem that machine will consider this words as important is also solved.

Also in our speeches the words which are two lettered and very big words are not also very useful. That is why we are also going to remove the words which have less that or equal to two letters and greater than 20 words.

After removing this words we are going to move to our next data preprocessing zone which is stemming the data.

STEMMING THE DATA

We are going to use the Porter Stemmer library of the ntlk package. First of all let us understand what does Stemming means ? Stemming means to convert the tokens into their root form. For example the word ‘flew’ and ‘fly’ means the same thing in English just the tense this changed. But for computer this two words are completely different words and there is no correlation between them. So What we are going to do with the help of this library is to convert all the words to their root form to make the work more easier for our model or machine.

CONVERTING THE LIST OF TOKENS TO STRING AGAIN

Since the time we tokenized our speeches, it was in the form of list of tokens but now dince most of the preprocessing part is done we are going to convert our data to strings again so that we can carry on with our furthur phase of the preprocessing part.
BUT THERE IS A VERY BIG ISSUE NOW ?

We collect the data in string format. We cleaned it , tokenized it , removed the unecessary words from it , converted those tokens to their to thier root forms and got the string back from those tokens but the issue is that no model works on string data.

VECTORIZING THE DATA

All the models works on some type of numeric data. So our next task will be to convert our data from string format to numeric format and for this purpose we are going to use the **TfidfVectorizer** class of **sklearn** library.

How does it works ??
Tfidf means Term Frequency Inverse Document Frequency. It take into account the number of time a particular word is repeated in a speech to calculate the importance of the speech

so this part is known as term frequency. But it also taken into account the number of speeches in which this word has shown up and this is known as document frequency. So it not only gives weightage to a word on the basis of a single speech but it also takes into account all the other speeches also to give the weightage to a particular word in the speech.

So a particular word gets high weightage gets higher weightage if its term frequency is high and document frequency is low. Since the importance of a word is directly proportional to term frequency and inversely proprotional to document frequency that is why this technique is known as term frequency inverse document frequency vectorizer.

So we will use this at this part of our project to convert our data from string format to numeric format.

Now our data is collected , cleaned , tokenized , again cleaned then stemmed and converted to string again and then finally got converted into numeric form. Now it is ready to be inputed into a model for clustering.

DATA CLUSTERING PART

We are going to use the K-means clustering to cluster our data. Why do we want to use K means clustering ?

Because as mentioned earlier that we do not require any labels for our clusters as it is an unsupervised clustering algorithm. It is very easy to implement and moreover we do not require any knowledge about the dataset. This method will automatically finds and detects any pattern in the data.

We have a very big issue now and that very issue is to select an optimal value of k if we do not select an optimal value of k then our clustering will not happen very good and it will not give good result so how to select a good value of k ? In order to select a good value of k there are various method and we are going to use two of them the first elbow method and the second method is Sillhouette score method.

In elbow method we try different values of k and calculate the distance metric in each case and when we plot this graph we get a graph which is declining and then becoming static at a position . This point at which the graph started becoming static is the optimal value of K. Though in our case we are not able to find the optimal value of k but we can see that the distance is decreasing which is a good indication for us.

In Sihoueete score method we calculate the sihouette score. This score tell that how near the data of a cluster is to the other data of cluster and how far the data of one cluster is from the data of another cluster. The ideal value of sihouette score is between -1 to 1. In our case we are getting a average value of sihoueete score at 5 which indicate that 5 is a good value of K.

Seeing both this graph we decided to apply the K means on our data with k = 5.

VISUALIZING THE DATA AFTER APPLYING THE KMEANS

Then we tried to plot the top most used words in each cluster. We also used wordcloud library to show the same information. In the wordcloud the word having higher weightage in the corpus is bigger in size while the word having less weightage is of low size.

FINDING THE PERSONALITY PROFILE OF PRESIDENTS

Then we found the positive sentiment , negative sentiment , compound sentiment in each speech using the **SentimentIntensityAnalyzer** class of the ntlk package. It uses some predefined rules and heuristices to determine this things.

We also calculated the **Confidence level** in each speech by finding the difference in the positive sentiment and negative sentiments.

GETTING RANKING FOR EACH PRESIDENT

Then we manually scraped a website to get the ranking of each of the president so that we can use it to predict the ranking of a speaker of a new given speech.

FINAL APPLICATION

The final application will take a new speech and determine the cluster in which the president belongs to and will also predict the ranking of the speaker of the speech by averaging the rankings of the top 5 nearest neighbors of the this speech.

THANK YOU