Topic Modeling U.S. Presidential Speeches: A Comparison of LDA and BERTopic on a Manually Collected Corpus

University of Salford MANCHESTER

BY

ODERA ISAAC NNAJI

@00801848

TO

DR ALI ALAMEER

FACULTY OF SCIENCE, ENGINEERING AND ENVIRONMENT

APRIL, 2025

TABLE OF CONTENT

ABSTRACT **4**

INTRODUCTION **5**

1.2. Problem Statement 6

1.3. Aim & Objectives 6

1.3.1 AIM 6

1.4 Significance of Study 7

1.5 Project Scope & Project Limitation 7

1.5.1 Scope 7

1.5.2 Limitations of Study 7

SYSTEM ARCHITECTURE DIAGRAM **8**

METHODOLOGY **9**

2.1. Dataset Acquisition 9

2.2. Dataset Overview 9

2.3. Exploratory Data Analysis (EDA) 9

2.3.1 Initial Data Loading & Inspection: 9

2.3.2 Data Inspection: 10

2.3.4 Data Cleaning and Formatting: 11

2.3.5 Temporal Distribution: 11

2.3.6 Transcript Length: 12

2.3.7 Presidential Distribution: 12

2.3.8 Average speech Length by President: 13

2.3.9 Word Frequency (Pre-preprocessing): 13

2.4. Data Cleaning 14

2.5. Text Preprocessing 14

2.6 N-gram Analysis and Word Cloud of our cleaned Text 16

IMPLEMENTATION **19**

3.0. Implementation in Cloud and Linux Virtual Environments **19**

3.1. Algorithmic Descriptions 19

3.1.1. Latent Dirichlet Allocation (LDA): 19

3.1.2 BERTopic: 20

3.2 Libraries Used **21**

3.3. Experimental Approach **23**

3.3.1 LDA Model Setup & Implementation: 23

3.3.2 BERTopic Model Setup & Implementation: 24

3.4. Validation Approach 26

3.5 Visualization of Results 27

1. LDA KEYWORD BAR CHART 28

2. BERTopic Visualization: 29
BERTopic Term Rank Visualization(visualize_term_rank()): 30
BERTopic visualize_hierarchy() Dendrogram: 30
BERTopic visualize_topics() Intertopic Distance Map: 31
Uniform Keyword Charts 32
RESULT ANALYSIS AND DISCUSSION **33**
4. 1. LDA Model Interpretation & Grouping 33
4.2. BERTopic Model Interpretation & Grouping 34
4.3. Comparative Analysis: LDA vs. BERTopic: 35
4.4 Keyword Comparison Based on Same/Similar Topic 37
4.5. Performance Comparison: Cloud vs. Local **40**
CONCLUSION AND RECOMMENDATION **41**
5. 1. Conclusion 41
5.2. Recommendation and Future Work: 42
5.3. Ethical, Legal, and Professional Considerations 43
REFERENCE **44**

ABSTRACT

This assessment report shows the entire Natural Language pipeline for Topic Modelling, from the initial process of data collections, data cleaning & preprocessing, as well as analyzing a corpus of US Presidential speeches using two different topic modeling methods based on the task requirements and I have chosen: BERTopic Algorithm & Latent Dirichlet Allocation (LDA).

The objective for this experiment is to identify, interpret, and compare the thematic relationships/structures found by these topic modelling techniques within this corpus. The corpus made up of about 1061 presidential speeches was collected and then it was passed through a preprocessing pipeline which includes HTML removal, basic cleaning, lemmatization using spaCy, stopword removal, and tokenization. After preprocessing, this cleaned text was then fed as input for both LDA (implemented via Gensim with 10 topics) and BERTopic (using a sentence transformer which is called “all-MiniLM-L6-v2 embedding”, as for dimensionality reduction i utilized UMAP, and KMeans clustering forced to 10 topics).

The results obtained, including topic keywords and as well as the visualizations generated (pyLDAvis for LDA, BERTopic's, both using t-SNE for mapping), were studied, compared and meaning was made out of these results. LDA was able to show us themes related to historical governance, different phases of foreign policy/conflict, economic policy, and general political topics and matters. BERTopic gave topics and themes that focused on the US governance, distinct foreign policy/conflict themes, national/world matters, and modern political communication styles. A comparative analysis highlighted differences in keyword coherence, thematic difference, and visualization, showing that while both methods offered valuable insights, BERTopic (visualized with t-SNE) provided clearer topic separation in its 2D map for this dataset, while LDA arguably offered a better thematic breakdown in certain areas like foreign policy eras. Finally, we will also highlight how running these models locally versus on cloud platforms (Google Colab) differs, as well as how the insight obtained from this task shows the importance of selecting the right topic modeling techniques and visualization methods based on your specific dataset characteristics and analytical goals would help you achieve good topic modelling systems.

INTRODUCTION

As of today, April 25, we generate billions or even trillions of unstructured data every single day. The world we live in now has turned into a data-driven one, where even data that was generated many years ago are now being repurposed to create useful products and valuable insights are also obtained from this data. Topic modeling is one of the ways we can make sense of unstructured data—data that doesn't follow any specific format. With NLP(natural language processing) techniques like topic modeling, we can take these kinds of data and draw meaning from them.

For example, some years ago, we had presidential speeches—just recorded and stored for historical sake. But now, we've seen that there's so much insight we can get from those same speeches. Instead of reading through the whole speech from every president on topics like the Iran issue or other political matters, we can simply gather all the manuscripts, feed them into a topic modeling system, and pull out useful information and patterns like identify recurring themes, categorize content, and summarize key topics. So, with topic modeling, we're able to turn large, unstructured texts into meaningful data that can help us understand trends and make better decisions.

Speaking about unstructured data, the thoughts and topics of former US presidents, captured in their official speeches, messages, and addresses, makes an interesting and unique corpus for understanding the trajectory of American history, political priorities, and societal concerns.

Topic modeling simply helps identify various, distinct, rare or frequent "topics" or "themes" that are present in a large corpus. These topics are represented as distributions over words or clusters as we call them, allowing researchers to spot patterns, or see from generated keywords the recurring topic, general theme and subjects from any large corpus. Topic modelling methods such as Latent Dirichlet Allocation (LDA) depend on probability, this works based on frequency of words appearing together, which is what we call Bag-of-Words. While new methods like BERTopic, which leverages the semantic understanding captured by large language models (like BERT) uses this to identify more details, relationship and exact them and not just generalizing relevant topics.

This report discusses the challenge of getting meaningful thematic insights from a manually collected corpus of 1061 U.S. Presidential speeches from the early 1790's to 2024. The primary research question guiding this investigation is: How effectively can Latent Dirichlet Allocation (LDA) and BERTopic identify distinct and semantically correct topics within this collected dataset of U.S. Presidential speeches? To answer this, I explore several secondary questions: What are the key

thematic similarities and differences revealed by each model? How do their underlying algorithmic mechanisms influence the resulting topic structures and interpretations? Which model, based on qualitative assessment of topic coherence and visualization clarity, provides more insightful results for understanding the evolution of Presidential discourse?

To conduct this comparative analysis, we talk about our corpus collection, preprocessing pipeline, highlight the use of both LDA and BERTopic , including the specific parameters and configurations chosen, ensuring identical preprocessed input for both models to facilitate a fair comparison. The experimental procedure, including the rationale for the chosen number of topics (k=10) and the qualitative validation approach, is outlined.

1.2. Problem Statement

The corpus of US Presidential speeches from across different centuries and covers a lot of domestic and international issues. Identifying the core themes, understanding how they shift across different presidencies and historical periods, and comparing how different presidents tackled/addressed different subjects . This project helps researchers and those interested to draw insight from different presidents ideology or approach to similar matters arising using modern NLP methods.

1.3. Aim & Objectives

1.3.1 AIM

I will perform topic modelling using an unstructured data set leveraging the unique abilities of unsupervised learning and NLP(natural language processing to derive clusters in form of topics from this dataset)

1.3.2 Objective

My major objectives of this project are:

1. To collect a unique dataset of US Presidential speeches suitable for topic modeling.
2. Clean & Preprocess the data. So it can be fed into our topic modelling models.
3. Apply Latent Dirichlet Allocation algorithm as well as BERTopic.
4. Interpret the topics generated by each model based on their keywords .
5. Compare the results and performance of both LDA and BERTopic, focusing on the semantic understanding of topics, generated keyword quality, and the insights from visualizations.
6. Document the process, including code implementation, results, and analysis, in this report.

7. Discuss the performance differences when running the models on Jupyter notebook(local machine) vs Google Co-Lab (cloud environments.)

1.4 Significance of Study

The significance of this project includes:

1. Making sense and finding trends, relationships from Historical Political Speeches: This project provides great help and insights to historians, political scientists, linguists, and the public seeking to understand the trends of American political thought and rhetoric.
2. Comparing different NLP Methodology (Traditional vs Deep Learning models):Topic modeling as a field has evolved and there are many needs for it. This task & report helps show the difference and uniqueness in traditional probabilistic models like LDA to newer techniques leveraging deep learning embeddings like BERTopic.

1.5 Project Scope & Project Limitation

1.5.1 Scope

This report covers certain scopes and will not go beyond that. Here they are:

1. Dataset: The task is centered on a curated corpus of 1061 U.S. Presidential speeches, messages, and addresses from the year 1790 to early 2024. It does not claim to have *all* Presidential communications but represents a minority sample of major public speeches.
2. Topic Modeling Methods: The study focuses specifically on comparing Latent Dirichlet Allocation (LDA), as implemented in the Gensim library, and BERTopic, using its pipeline with specific configurations (Sentence-BERT embeddings, UMAP dimensionality reduction, KMeans clustering focused on 10 topics). Other topic modeling algorithms (e.g., NMF, LSA, other embedding/clustering combinations within BERTopic) are outside the scope of this report.

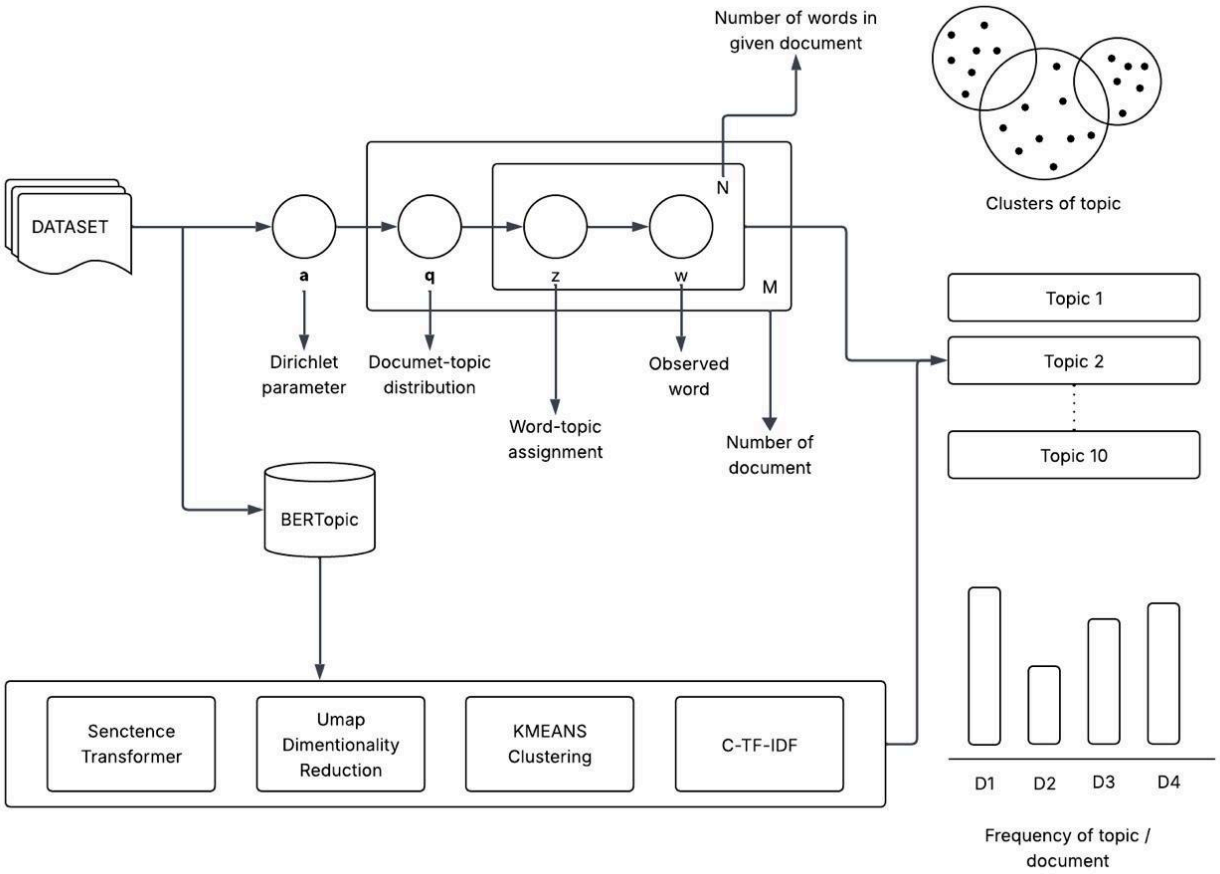
1.5.2 Limitations of Study

While this study provides valuable comparative insights, several limitations should be acknowledged:

1. Dataset is limited to just a minority of presidential speeches and addresses made between a period of time and not all US presidential speeches.
2. Applying the same preprocessing like I did for LDA i.e (lemmatization, stopword removal) may or may not have prevented BERTopic from giving it best performance.
3. Model Generalization: My findings on the performance of LDA and BERTopic and the specific topics identified are specific to this corpus of Presidential speeches and the chosen model configurations. Results will differ on other datasets or with different parameter settings.

SYSTEM ARCHITECTURE DIAGRAM

This diagram design presents and effectively summarizes my system architecture. It shows how the dataset(presidential speeches) is sent as input and passed through the different topic models (LDA and BERTopic and the outputs are clusters of topics.



METHODOLOGY

2.1. Dataset Acquisition

For this project, I made use of a corpus that contains transcripts of speeches delivered by Presidents of the United States. Based on the project requirements, this corpus unlike readily available datasets, was gotten from an online archives which is The American Presidency Project (presidency.ucsb.edu) hosted by the University of California, Santa Barbara, and the Miller Center at the University of Virginia (millercenter.org). The documents in this corpus were gathered within different time period starting from around 1790 up to 2024, which provides us with a vast theme, different vocabulary as some of the english vocabulary in the 1700's compared to the 2000's are different, as well as the problems and issues faced and spoken about during this time frame.

2.2. Dataset Overview

This corpus contains 1061 documents (speeches). Key fields in this datasets include:

- Transcript: Original speech transcript.
- President: Name of president who delivered the speech.
- Date: Day, Month, Year that the speech was given.
- Title: The title of the speech i.e 'Address to congress', 'Independence day speech'.

2.3. Exploratory Data Analysis (EDA)

Before we start modeling, several EDA steps were performed to understand the dataset's characteristics:

2.3.1 Initial Data Loading & Inspection:

This shows our data, the dataset fields, and just an initial feel of what our dataset contains.

```
import requests
import json

# API Endpoint
endpoint = "https://api.millercenter.org/speeches"

# Initialize variables
all_speeches = []
params = {}

# Pagination loop
while True:
    response = requests.get(endpoint, params=params)
    if response.status_code != 200:
        print("Error:", response.status_code, response.text)
        break

    data = response.json()
    all_speeches.extend(data.get("Items", []))  # Append speeches

    # Check if more data is available
    if "LastEvaluatedKey" in data:
        params = {"LastEvaluatedKey": data["LastEvaluatedKey"]["doc_name"]}
    else:
        break  # No more data to fetch

# Save data to a JSON file (optional)
with open("presidential_speeches.json", "w", encoding="utf-8") as f:
    json.dump(all_speeches, f, indent=4)

print(f"Downloaded {len(all_speeches)} speeches.")
```

```
Downloaded 1061 speeches.
time: 1min 53s (started: 2025-04-15 18:32:03 +01:00)
```

2.3.2 Data Inspection:

Using the ".head()" function we can see what our dataset columns and rows look like.

```
df = pd.DataFrame(all_speeches)
# Display first few rows
df.head(10)
```

	doc_name	date	transcript	president	title
0	january-22-1807-special-message-congress-burr-...	1807-01-22	TO THE SENATE AND HOUSE OF REPRESENTATIVES OF ...	Thomas Jefferson	January 22, 1807: Special Message to Congress ...
1	may-25-1813-message-special-congressional-sess...	1813-05-25	Fellow-Citizens of the Senate and of the House...	James Madison	May 25, 1813: Message on the Special Congressi...
2	april-2-1917-address-congress-requesting-decla...	1917-04-02	I have called the Congress into extraordinary ...	Woodrow Wilson	April 2, 1917: Address to Congress Requesting ...
3	april-10-1975-address-us-foreign-policy	1975-04-10	Mr. Speaker, Mr. President, distinguished gues...	Gerald Ford	April 10, 1975: Address on U.S. Foreign Policy
4	july-6-1848-message-regarding-treaty-guadalupe...	1848-07-06	To the House of Representatives of the United ...	James K. Polk	July 6, 1848: Message Regarding the Treaty of ...
5	march-19-2008-remarks-war-terror	2008-03-19	Thank you all. Deputy Secretary England, thank...	George W. Bush	March 19, 2008: Remarks on the War on Terror
6	june-25-1965-remarks-20th-anniversary-un-charter	1965-06-25	\r\n\r\nMr. President, Mr. Secretary General,...	Lyndon B. Johnson	June 25, 1965: Remarks on the 20th Anniversary...
7	june-24-1938-fireside-chat-13-purging-democrat...	1938-06-24	I think the American public and the American n...	Franklin D. Roosevelt	June 24, 1938: Fireside Chat 13: On Purging th...
8	december-2-1823-seventh-annual-message-monroe-...	1823-12-02	Fellow Citizens of the Senate and House of Rep...	James Monroe	December 2, 1823: Seventh Annual Message (Monr...
9	january-9-1961-city-upon-hill-speech	1961-01-09	I have welcomed this opportunity to address th...	John F. Kennedy	January 9, 1961: "City Upon a Hill" Speech

```
time: 54.3 ms (started: 2025-04-15 18:33:57 +01:00)
```

- **Data Structure:** My dataset contains 1061 entries, which is made up of textual transcripts. Initial df.info() confirmed data types, with transcript, president, and title being objects (strings) and date requiring conversion.

```
# Generate descriptive statistics to understand data distribution
df.describe(include='all')
```

	doc_name	date	transcript	president	title
count	1061	1061	1061	1061	1061
unique	1061	1038	1060	45	1055
top	january-22-1807-special-message-congress-burr-...	1920-07-22	By the President of the United States of Ameri...	Lyndon B. Johnson	December 6, 1892: Fourth Annual Message
freq	1	5	2	71	2

```
time: 27.3 ms (started: 2025-04-15 18:33:57 +01:00)
```

```
# 1. Get a quick overview of the data typ
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1061 entries, 0 to 1060
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   doc_name    1061 non-null   object
 1   date        1061 non-null   object
 2   transcript  1061 non-null   object
 3   president   1061 non-null   object
 4   title       1061 non-null   object
dtypes: object(5)
memory usage: 41.6+ KB
```

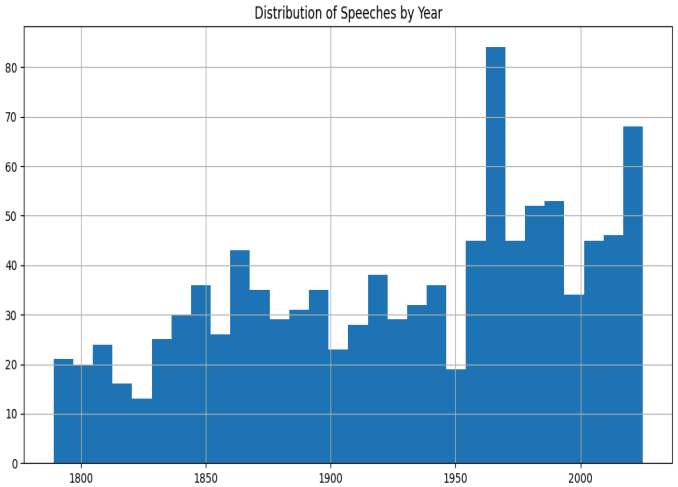
2.3.4 Data Cleaning and Formatting:

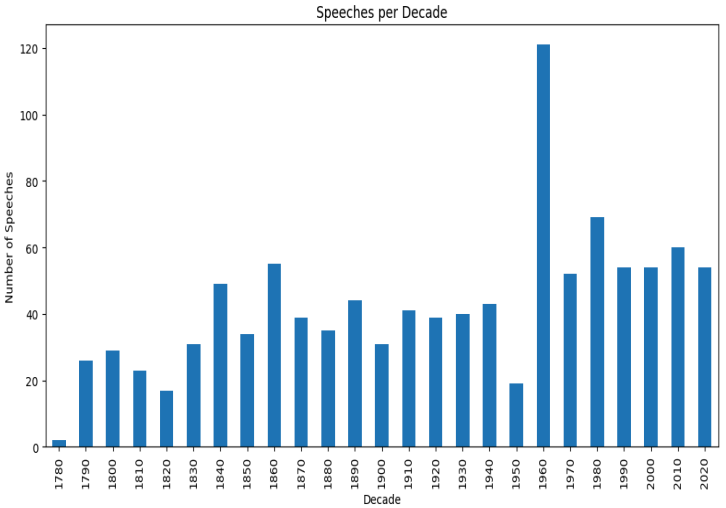
I had to clean & convert the date column to a standard datetime format using pandas to_datetime with errors='coerce'. Initial inspection using df.info() and df.describe() showed us the data types, missing values, and basic distributions.

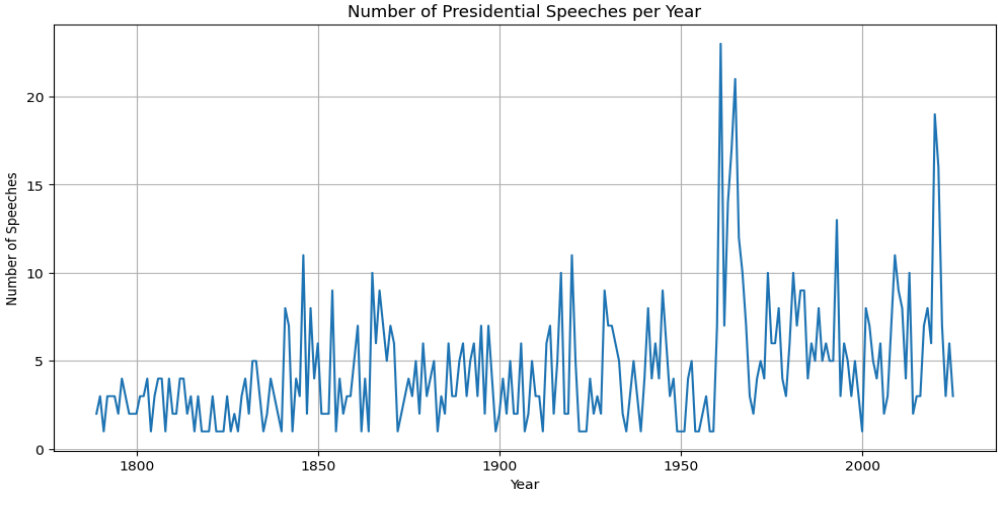
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1061 entries, 0 to 1060
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   doc_name    1061 non-null   object
 1   date        1061 non-null   datetime64[ns, UTC]
 2   transcript  1061 non-null   object
 3   president   1061 non-null   object
 4   title       1061 non-null   object
 5   year        1061 non-null   int32
dtypes: datetime64[ns, UTC](1), int32(1), object(4)
memory usage: 45.7+ KB
None
Number of null dates = 0
```

2.3.5 Temporal Distribution:

I used a line plot and bar charts to visualize the number of speeches per year and per decade showed fluctuations over time. This revealed periods of higher and lower recorded speech activity, potentially reflecting historical events (wars, crises) or changes in Presidential communication styles and record-keeping.

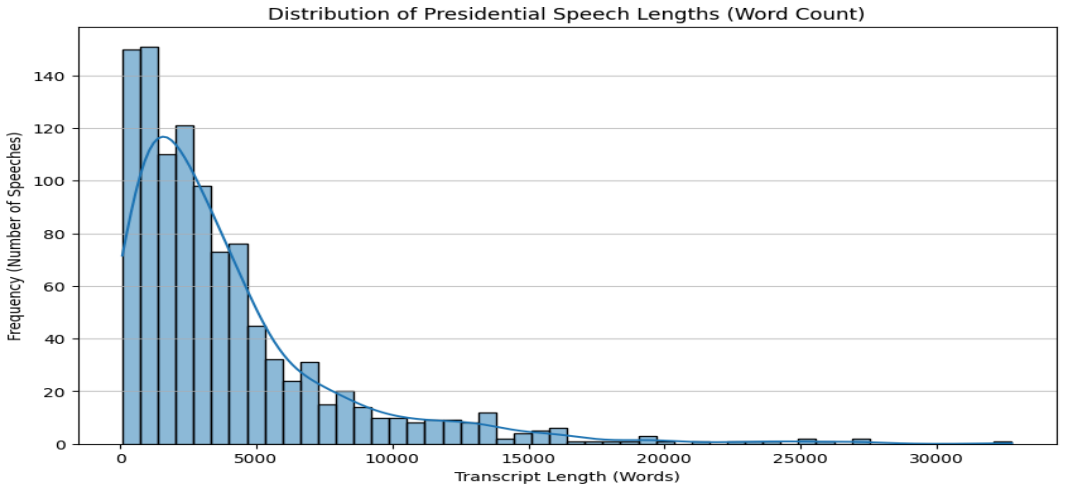






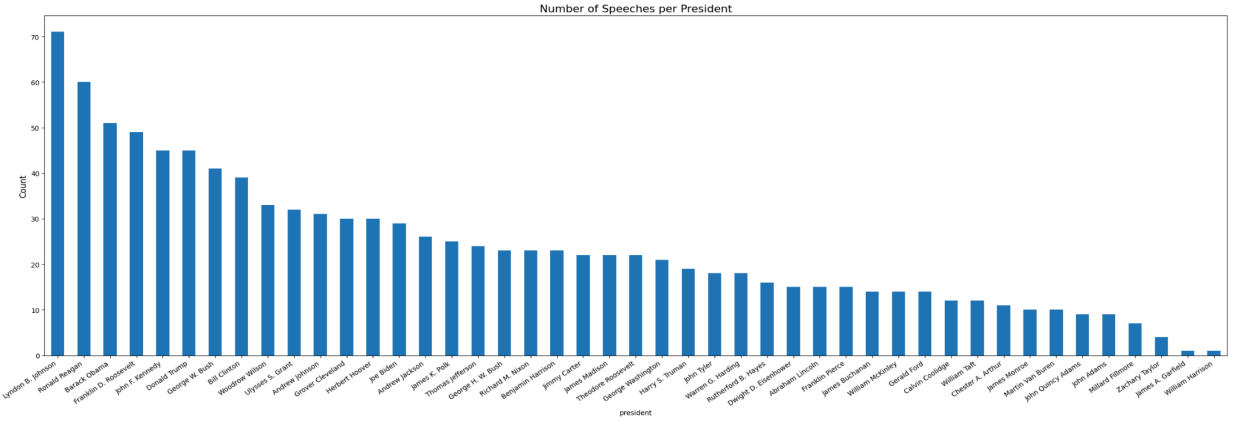
2.3.6 Transcript Length:

I calculated the word count for each transcript (df['transcript_length']). Descriptive statistics and a histogram showed different ranges of speech lengths, with a likely right-skewed distribution (many shorter speeches, fewer very long ones). I was able to find the shortest and longest speeches.



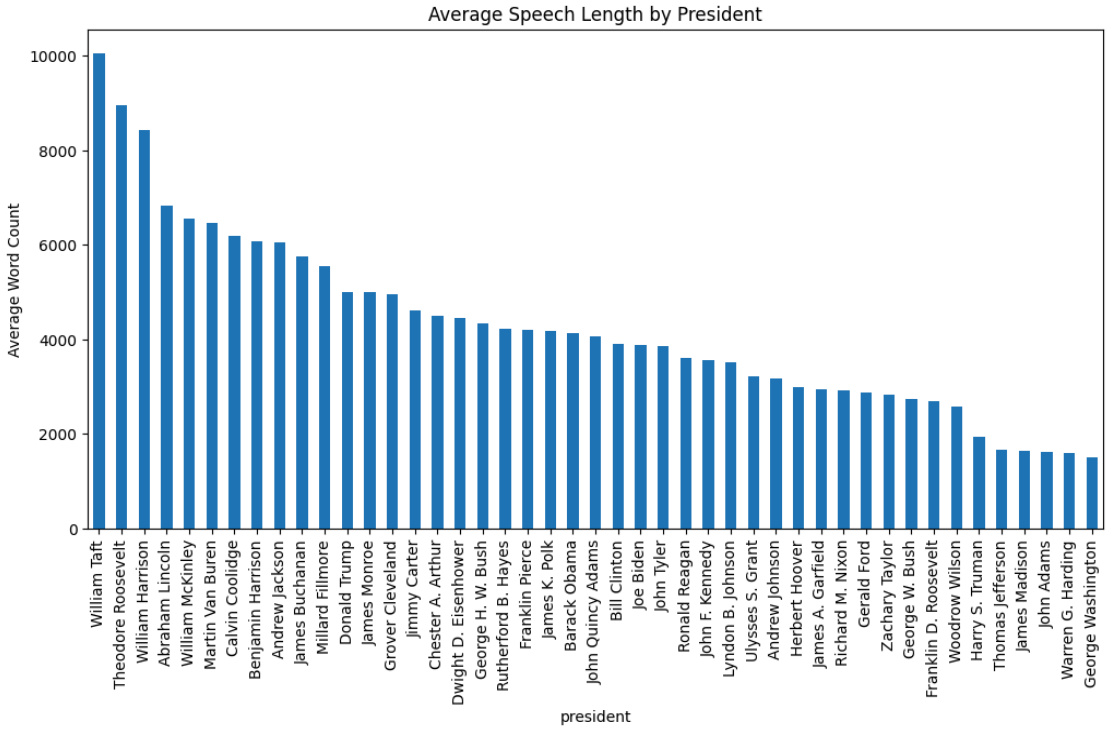
2.3.7 Presidential Distribution:

I used a bar chart to show the number of speeches attributed to each President in the dataset. This showed so many variations, with some Presidents being much more heavily represented than others.



2.3.8 Average speech Length by President:

I also found the average speech length by individual presidents so as to gain an insight on which president, the year and what our topic will mostly be centred around.



2.3.9 Word Frequency (Pre-preprocessing):

For topic modelling, word frequency matters a lot, so we have to carefully look at the common and least common words we have in our corpus, in order decide what to keep or remove .English stop words (like 'the', 'of', 'and') and potentially indicated domain-specific frequent terms (like 'government', 'states', 'people'). Stopwords made up a significant percentage of the total word count, showing me the need for their removal, especially for LDA.

```
Top 60 Common Words:
[17]:

[('the', 293966),
 ('of', 197516),
 ('to', 139484),
 ('and', 130959),
 ('in', 81315),
 ('a', 66645),
 ('that', 58901),
 ('is', 41163),
 ('for', 39045),
 ('be', 38795),
 ('our', 34817),
 ('I', 34522),
 ('have', 31146),
 ('by', 28363),
 ('we', 27317),
 ('as', 26001),
 ('with', 25496),
 ('this', 24732),
 ('it', 24668),
 ('are', 22550),
 ('not', 22368),
 ('will', 21987),
 ('which', 21934),
 ('on', 20886),
 ('The', 18876),
```

```
30 Least Common Words:
[18]:

[('1,449', 1),
 ('1,730', 1),
 ('3,179.', 1),
 ('36,551', 1),
 ('20,126', 1),
 ('4,845', 1),
 ('11,580', 1),
 ('33,838.', 1),
 ('9,104', 1),
 ('232,229', 1),
 ('1872.It', 1),
 ('$30,480,000', 1),
 ('year.THE', 1),
 ('CENSUS.', 1),
 ('publication.The', 1),
 ('century.EDUCATION', 1),
 ('evidences,', 1),
 ('Congress.TERRITORIES', 1),
 ('.Affairs', 1),
 ('(Utah)', 1),
 ('polygamy,Since', 1),
 ('embellishments', 1),
 ('authorities.AGRICULTURE', 1),
 ('horticultural,', 1),
 ('statistical,', 1),
 ('entomological,', 1),
 ('chemical--and', 1),
 ('pursuits.The', 1),
```

2.4. Data Cleaning

Based on the EDA, the following cleaning steps were applied:

1. Column Selection: I kept all essential columns (transcript, president, date).
2. Date Conversion: As mentioned above, our dates were converted to a consistent datetime format.
3. HTML Tag Removal: A function (remove_html_tags) using regular expressions (re.sub(r'<[^>]+>', '', text)) was applied to the transcript column to remove any HTML markup that can b found in archived texts.

```python
# 2. Function to remove HTML tags
def remove_html_tags(text):
    clean_text = re.sub(r'<[^>]+>', '', text)
    return clean_text
```

2.5. Text Preprocessing

For text preprocessing I made a decision to apply the same text preprocessing pipeline for our text regardless of our model choice (LDA or BERTopic). While BERTopic can handle less processed text effectively because of its contextual embeddings, but for the sake of fair comparison using the same input allows for a more direct analysis of the topic modeling algorithms themselves, minimizing the variableness of different preprocessing steps which can actually influence results.

I used the following preprocessing pipeline:

1. **Basic Text Cleaning**:
    - I removed special characters & numbers using regex, i kept only meaningful characters and whitespace.
    - I converted text to lowercase: text.lower().

```python
# Step 1: Data Cleaning
# Keep only relevant columns
df = df[['transcript', 'president', 'date']]  # Keep pres

# Step 2: Text Preprocessing
def clean_text(text):
    # Remove special characters and numbers
    text = re.sub(r'[^a-zA-Z\s]', '', text, re.I|re.A)
    # Convert to lowercase
    text = text.lower()
    # Remove extra whitespace
    text = re.sub('\s+', ' ', text)
    return text

df['clean_text'] = df['transcript'].apply(clean_text)

# Step 3: Word Cloud Visualization
# Join all texts
all_text = ' '.join(df['clean_text'])

clean_text(df['transcript'][0])
```

2. **Lemmatization:**

   - I chose lemmatization over stemming so as to properly reduce words, while preserving better interpretability (e.g., "meeting" -> "meeting" vs. "meet").
   - I made use of the spaCy library (en_core_web_sm model) for efficient and accurate lemmatization. The function processed the text with spaCy.

3. **Tokenization:**

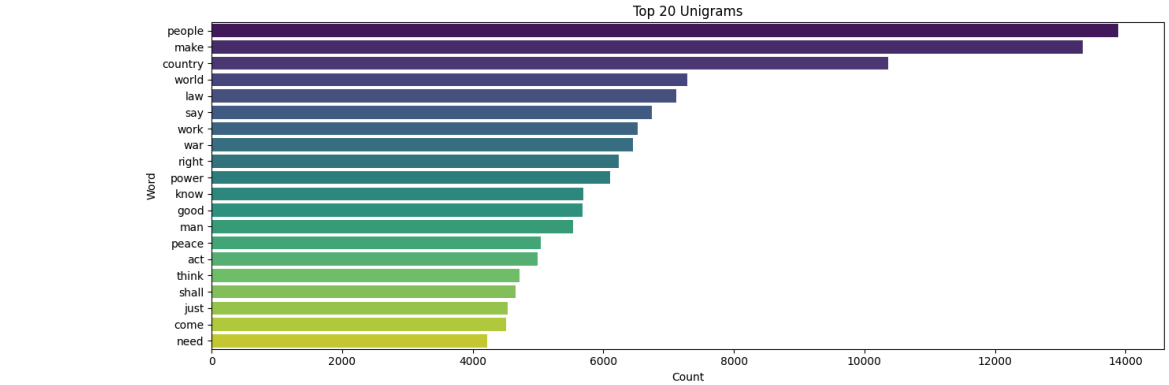
   - Even though lemmatization was performed first using spaCy, for consistency with certain modeling library inputs (like Gensim) and for analysis (like N-grams in EDA), we performed tokenization as well.

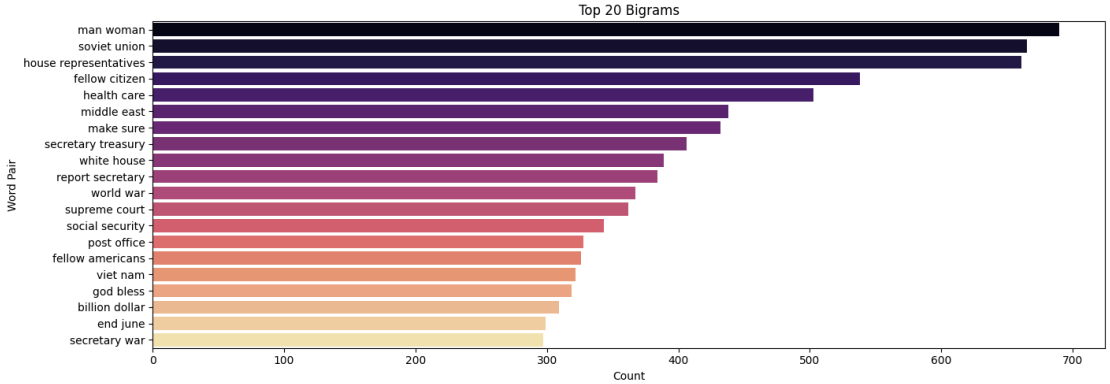
```
# Load spaCy model for lemmatization
nlp = spacy.load('en_core_web_sm')

# 1. Define a custom stop word list - extending the default English list
custom_stop_words = list(CountVectorizer(stop_words='english').get_stop_words()) # Start
custom_stop_words.extend([
    'states', 'united', 'government', 'congress', 'president', 'year', 'years', 'time',
])
custom_stop_words_list = list(custom_stop_words)  # Convert set to list for CountVectori

# 2. Function to remove HTML tags
def remove_html_tags(text):
    clean_text = re.sub(r'<[^>]+>', '', text)
    return clean_text

# 3. Lemmatize text
def lemmatize_text(text):
    doc = nlp(text)
    lemmatized_words = [token.lemma_ for token in doc]
    return " ".join(lemmatized_words)

# 4. Sentence to words
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc=True r

# 5. Remove stopwords
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc))
             if word not in custom_stop_words_list] for doc in texts]

# Assuming df is your DataFrame with a 'text' column containing the raw text
# Apply HTML tag removal first to the transcript column
df['clean_text'] = df['transcript'].apply(remove_html_tags)  # Process raw transcripts

# Apply lemmatization
df['clean_text'] = df['clean_text'].apply(lemmatize_text)

# Convert to list
data = df['clean_text'].values.tolist()

# Tokenization
data_words = list(sent_to_words(data))

# Remove stopwords FIRST PASS
data_words = remove_stopwords(data_words)
```

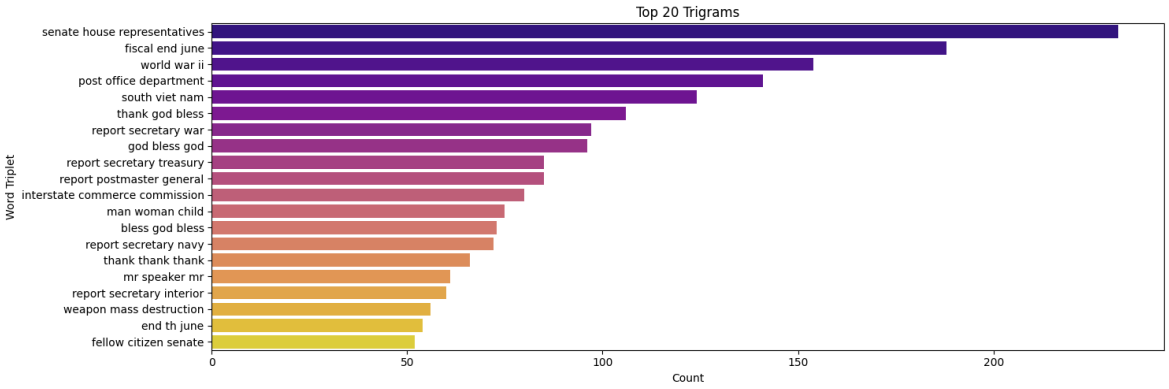
4. **Stopword Removal (remove_stopwords function):**
    - Rationale: A bid for me to remove high occuring words that provide little semantic value for seperating between topics.
    - Method:I started by applying the standard English stopword list from nltk.corpus.stopwords.
    - Custom List: i had to also create an extended list with domain-specific terms identified during EDA as being highly frequent and likely to appear across many topics in Presidential speeches: ['states', 'united', 'government', 'congress', 'president', 'year', 'years', 'time', 'great', 'new', 'public', 'state', 'nation', 'american', 'america']. The final custom_stop_words_list included both standard and custom words.

2.6 N-gram Analysis and Word Cloud of our cleaned Text

From the N-gram analysis below we can now see the top words (Uni, Bi, & Tri gram) as well as a word cloud visualization of these words.







Word Cloud representation of our most frequent words.

N-Gram Summary:

Rank	Top Unigrams	Count	Top Bigrams	Count	Top Trigrams	Count
1	people	13,894	man woman	690	senate house representatives	232
2	make	13,350	soviet union	665	fiscal end june	188
3	country	10,362	house representatives	661	world war ii	154
4	world	7,287	fellow citizen	538	post office department	141
5	law	7,119	health care	503	south viet nam	124
6	say	6,747	middle east	438	thank god bless	106
7	work	6,523	make sure	432	report secretary war	97
8	war	6,452	secretary treasury	406	god bless god	96
9	right	6,232	white house	389	report secretary treasury	85
10	power	6,109	report secretary	384	report postmaster general	85

2.7 Final Processed Data for Models

- LDA: After preprocessing the df['clean_text'] (lemmatized strings) column i fed to the sklearn.feature_extraction.text.CountVectorizer which handled tokenization based on the token_pattern and removed stopwords using custom_stop_words_list to create the document-term matrix (X_lda). This matrix was then converted to the Gensim corpus format.
- BERTopic: The same df['clean_text'] column was still used as the input (docs_for_bertopic). BERTopic's internal pipeline then handles embedding. For keyword generation, its internal CountVectorizer was configured with ngram_range=(1,3) and standard English stopwords (applied *after* embedding/clustering to refine topic representations).

This extensive preprocessing helped to make sure that both models received clean, lemmatized text, free from common and domain-specific noise words, providing a good input for topic extraction.

IMPLEMENTATION

3.0. Implementation in Cloud and Linux Virtual Environments

For this topic modeling process, we need to have an environment in which we can perform these tasks like data preparation, model training, and visualization. All this was done using Python and several libraries. My implementation was done in two distinct environments: a cloud-based platform (Google Colab) and a local machine running Linux using a virtual environment (done via Anaconda Jupyter).

3.1. Algorithmic Descriptions

The two distinct topic modeling algorithms i used were:

3.1.1. Latent Dirichlet Allocation (LDA):

- **Concept:** LDA is a generative probabilistic model for collections of discrete data, such as text corpora. Developed by Blei, Ng, and Jordan (2003), It takes documents as random combinations of hidden topics, and each topic is made up of a group of a group of related words.
- **How it works:** LDA believes that each speech document was created through a process where you first pick topics (based on a Dirichlet distribution, α). Then, for each word in the document, you choose a topic from that mix, and finally, you pick a word from the vocabulary based on the chosen topic (drawn from another Dirichlet prior, β). LDA tries to return this process by means of statistical inference mainly using Gibbs sampling or most times variational inference. This then helps to discover hiding topic structures i.e the entire document-topic distributions as well as the topic words from our corpus
- **Input Data:** LDA works based on the same principle as Bag-of-Words, which is a document-term matrix that doesn't care about word order, all that matters is word frequencies. This is why good preprocessing matters, things like removing stopwords and lemmatization/stemming are needed and helps create our unique and rich vocabulary.
- **Output:** LDA outputs are probability groups of keywords that are similar and related which most certainly or probably belong to a certain theme or topic.

3.1.2 BERTopic:

- **Concept:** Unlike LDA, BERTopic is a recent topic modeling technique developed by Grootendorst. It is built on large pre-trained transformer models (like BERT or Sentence-BERT), dimensionality reduction is applied and you perform clustering to group related topics.
- **How it works:** BERTopic pipeline involves some steps:
    1. *Document Embedding:* Each document is converted into a dense number (embedding) using a transformer model (e.g., all-MiniLM-L6-v2). These embeddings capture semantic context, meaning similar documents are placed closer together in the embedding space.
    2. *Dimensionality Reduction:* Bertopic gives us large embeddings and we use UMAP to reduce these dimensions in order to make clustering possible and good while still preserving essential structure.
    3. *Clustering:* Documents are clustered in the reduced embedding space using algorithms like HDBSCAN (default, density-based, finds variable clusters and outliers) or KMeans (requires pre-specifying the number of clusters). Documents within the same cluster are said or assumed to share a common topic.
    4. *Topic Representation:* Keywords for each topic (cluster) are obtained using TF-IDF. This method helps us to compare word frequencies within a topic cluster versus its frequency across the entire corpus, finding words that are particularly representative of that specific topic. N-grams can also be considered if need be.
- **Input Data:** You can feed both raw or mildly processed text due to contextual embeddings, but it can also work with preprocessed cleaned text.
- **Output Data:** The output is usually keywords with c-TF-IDF scores for each topic. It sometimes is a calculated document-topic probability distribution.

3.2 Libraries Used

I installed and also imported some of the following libraries for the course of this project and i have grouped them by functionality with justifications:

Category	Library	Version	Usage	Reasons for Selection
Core Data Handling	pandas	2.2.4	Dataframe manipulation	For tabular data operations.
	numpy	1.26.4	Numerical computations and array operations.	Great for handling numerical data.
Visualization	matplotlib	-	Creating visualizations.	Used for plotting.
	seaborn	-	Statistical data visualization.	it simplifies complex visualizations.
	plotly	6.0.1	Interactive visualizations.	Enables dynamic and interactive plots.
Text Processing	nltk	3.9.1	Natural language processing tasks (e.g., stopwords).	Standard NLP toolkit.
	textblob	0.19.0	Sentiment analysis and text processing.	Simplifies sentiment analysis.
	spacy	-	Advanced NLP tasks (e.g., lemmatization).	NLP library with pre-trained models.
	gensim	4.3.3	Topic modeling (LDA) and document similarity.	Great for unsupervised topic modeling.
	pyLDAvis	3.4.1	Visualizing LDA topic models.	Provides interactive topic model visualizations.
Web/API	requests	2.32.3	Fetching data from APIs	Simplifies HTTP requests and handling responses.
Utility	re (regex)	2024.11.6	Text cleaning and pattern matching.	Used for regex operations in Python.
	collections	-	Data structures (e.g., Counter).	Provides specialized container data types.
	json	-	Handling JSON data.	Great for JSON serialization/deserialization.

Topic Modeling	BERTopic	-	Advanced topic modeling with transformer embeddings.	Combines transformer embeddings with clustering for modern topic modeling.
	umap-learn	-	Dimensionality reduction for BERTopic.	Good for visualizing high-dimensional data.
	scikit-learn	1.4.2	Clustering (KMeans) and machine learning utilities.	Versatile ML library with efficient clustering algorithms.
Miscellaneous	ipython-autotime	0.3.2	Tracking execution time of Jupyter/colab notebook cells.	Helps in performance monitoring.
	joblib	1.4.0	Parallel computing and caching.	Optimizes performance for repetitive tasks.
	tqdm	4.67.1	Progress bars for loops.	

```
# Data Cleaning
import numpy as np
import json
import re
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# Topic Modeling Prep
import gensim
from gensim.utils import simple_preprocess
from gensim import models, matutils
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
!pip  install textblob
from textblob import TextBlob
```

```
from sklearn.feature_extraction.text import CountVectorizer
import spacy
import random
from collections import Counter

!pip install pyLDAvis
import pyLDAvis.gensim_models as gensimvis
import pyLDAvis.gensim_models
import pyLDAvis
from gensim.corpora import Dictionary
```

3.3. Experimental Approach

3.3.1 LDA Model Setup & Implementation:

- **Input Data:** I fed in a list of strings from the preprocessed df['clean_text'] (lemmatized text).
- **Vectorization:** I then used "CountVectorizer" with "stop_words=custom_stop_words_list", and an ngram range of (1, 1) to create a document-term matrix focusing on meaningful unigrams.
- **Gensim Prep:** The vocabulary from the vectorizer was then used to create a id2word mapping, and the sparse matrix (X_lda) was converted using "matutils.Sparse2Corpus" to form the corpus.
- **Model Training:** Now, I trained "gensim.models.LdaMulticore" with the following parameters: 10 number of topics and a random state of 100, and "per_word_topics=True". Ten numbers of topics was chosen as a common starting point, the same as the number of topics for BERTopic KMeans setup. "random_state=100" helps me ensure reproducibility. "passes=10" provides a balance between convergence and training time. LdaMulticore was used to leverage multiple CPU cores.

Here see the output after LDA training. It shows discovered topic and the embedding vectors

```
--- LDA Model Training ---
Document-term matrix shape: (1061, 29393)

--- Discovered Topics ---

Topic 0:
0.008*"law" + 0.008*"make" + 0.005*"country" + 0.005*"shall" + 0.005*"service" + 0.004*"act" + 0.004*"department" + 0.004*"increase" + 0.004*"general" + 0.004*"present"

Topic 1:
0.011*"people" + 0.010*"world" + 0.008*"peace" + 0.008*"war" + 0.007*"make" + 0.006*"country" + 0.005*"say" + 0.005*"man" + 0.005*"know" + 0.005*"freedom"

Topic 2:
0.016*"world" + 0.010*"peace" + 0.009*"people" + 0.007*"war" + 0.006*"soviet" + 0.006*"make" + 0.006*"freedom" + 0.006*"country" + 0.005*"force" + 0.005*"free"

Topic 3:
0.008*"make" + 0.007*"power" + 0.007*"country" + 0.006*"law" + 0.005*"people" + 0.005*"duty" + 0.005*"act" + 0.005*"constitution" + 0.005*"present" + 0.004*"subject"

Topic 4:
0.011*"tariff" + 0.009*"country" + 0.007*"increase" + 0.007*"make" + 0.007*"line" + 0.006*"duty" + 0.005*"rate" + 0.005*"article" + 0.005*"schedule" + 0.004*"product"

Topic 5:
0.009*"people" + 0.006*"federal" + 0.006*"country" + 0.006*"make" + 0.006*"world" + 0.006*"economic" + 0.005*"national" + 0.005*"shall" + 0.004*"business" + 0.004*"power"

Topic 6:
0.013*"man" + 0.010*"people" + 0.009*"make" + 0.006*"think" + 0.005*"war" + 0.005*"world" + 0.005*"right" + 0.005*"say" + 0.005*"country" + 0.004*"come"

Topic 7:
0.014*"applause" + 0.012*"people" + 0.010*"make" + 0.010*"work" + 0.007*"know" + 0.007*"job" + 0.007*"country" + 0.006*"americans" + 0.006*"just" + 0.006*"say"

Topic 8:
0.016*"people" + 0.013*"say" + 0.011*"think" + 0.010*"know" + 0.009*"want" + 0.008*"make" + 0.007*"country" + 0.007*"just" + 0.006*"work" + 0.006*"come"

Topic 9:
0.009*"people" + 0.009*"tax" + 0.009*"make" + 0.008*"program" + 0.007*"increase" + 0.006*"country" + 0.006*"work" + 0.005*"federal" + 0.005*"billion" + 0.005*"think"
time: 3min 24s (started: 2025-04-15 18:55:03 +01:00)
```

3.3.2 BERTopic Model Setup & Implementation:

- **Input Data:** I used the *exact same* list of strings after preprocessing “df['clean_text']” was fed as “docs_for_bertopic”.
- **Embedding:** I used BERTopic's default all-MiniLM-L6-v2 SentenceTransformer model to generate document embeddings.
- **Dimensionality Reduction:** I configured UMAP to “n_neighbors=15”, “min_dist=0.0”, “metric='cosine'”, and “random_state=100” to match LDA seed for comparability in stochastic processes. The default number of components was used, providing space for better clustering.
- **Clustering:** With me aiming for a fair and direct comparison with LDA's output size, “sklearn.cluster.KMeans” was explicitly used instead of the default HDBSCAN. It was configured with n_clusters=10 (matching the LDA num_topics), random_state=100, and n_init='auto'. This forced the documents into 10 distinct topic clusters.
- **Topic Representation:** A CountVectorizer with ngram_range=(1, 3) and standard English stop_words was used on my bert vectorizer_model parameter. This helps me define how keywords (unigrams, bigrams, trigrams) are generated using the “c-TF-IDF algorithm” after clustering.

- **Model Training:** BERTopic was trained with the above components and fitted using "topic_model_bertopic.fit_transform(docs_for_bertopic)". In order to help with some visualization I set "calculate_probabilities=True".

BERTopic Training

```
# !pip install bertopic umap-learn hdbscan scikit-learn sentence-transformers plotly

import pandas as pd
from bertopic import BERTopic
from umap import UMAP
# We will use KMeans
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import CountVectorizer


if 'clean_text' not in df.columns:
     raise ValueError("DataFrame missing the 'clean_text' column with lemmatized text.")

print("Preparing data for BERTopic...")
docs_for_bertopic = df['clean_text'].tolist()

# Check data
if not docs_for_bertopic or not isinstance(docs_for_bertopic[0], str):
     raise ValueError("Input 'docs_for_bertopic' is not a valid list of strings.")

print(f"Number of documents for BERTopic: {len(docs_for_bertopic)}")
print("Sample document (first 500 chars):")
print(docs_for_bertopic[0][:500])

# --- Configure BERTopic Components ---
print("\nSetting up BERTopic components...")

# Dimensionality Reduction Model (match LDA random state)
umap_model_bert = UMAP(n_neighbors=15,
                       min_dist=0.0,
                       metric='cosine',
                       random_state=100) # Match LDA random state

# Clustering Model (Using KMeans like the teacher, targeting 10 topics like LDA)
num_target_topics = 10
kmeans_model_bert = KMeans(n_clusters=num_target_topics,
                           random_state=100,
                           n_init='auto') # Set n_init explicitly

# Keyword Extraction Model (Using n_gram_range)
vectorizer_model_bert = CountVectorizer(stop_words="english", ngram_range=(1, 3))
```

```python
# Embedding Model (Using BERTopic's default: "all-MiniLM-L6-v2")

# Initialize and Fit BERTopic Model
print(f"\nInitializing BERTopic model with KMeans (k={num_target_topics})...")
topic_model_bertopic = BERTopic(
                    umap_model=umap_model_bert,
                    hdbscan_model=kmeans_model_bert, # Pass the KMeans model here
                    vectorizer_model=vectorizer_model_bert,
                    language="english",
                    calculate_probabilities=True,
                    verbose=True
                )

print("\nFitting BERTopic model... This can take some time.")
topics_bert, probs_bert = topic_model_bertopic.fit_transform(docs_for_bertopic)

print("\nBERTopic model fitting complete.")

# Display Basic Results
# With KMeans, there should be exactly num_target_topics (e.g., 10) and no Topic -1
topic_info_bert = topic_model_bertopic.get_topic_info()
print("\nBERTopic Model Info (using KMeans):")
print(topic_info_bert)

print("\nKeywords for first few topics found:")
# Print top 5 topics found (should be 0-4 if num_target_topics=10)
num_topics_to_show = min(5, len(topic_info_bert))
for i in range(num_topics_to_show):
    topic_id = topic_info_bert.iloc[i]['Topic']
    print(f"Topic {topic_id}: {topic_model_bertopic.get_topic(topic_id)}")

# Add topic predictions to your DataFrame
df['bertopic_topic'] = topics_bert
print("\nAdded BERTopic topic assignments to DataFrame column 'bertopic_topic'.")
print(df[['clean_text', 'bertopic_topic']].head())
```

3.4. Validation Approach

Since this project focused on exploring themes in historical text and comparing how different topic modeling methods work, I didn't rely heavily on numbers or technical metrics like *perplexity* or *topic coherence scores* (such as C_v or UMass). While these metrics are useful, they don't always explain how meaningful generated topics are to humans.

Therefore, i focused on the following qualitative evaluations, to help us presents topic in a way humans can understand:

- **Semantic Coherence:** We checked if the top keywords in each topic actually made sense together and pointed to a clear, unified theme.
- **Interpretability:** We considered how easy it was to assign a meaningful label to each topic based on its keywords and the documents tied to it.

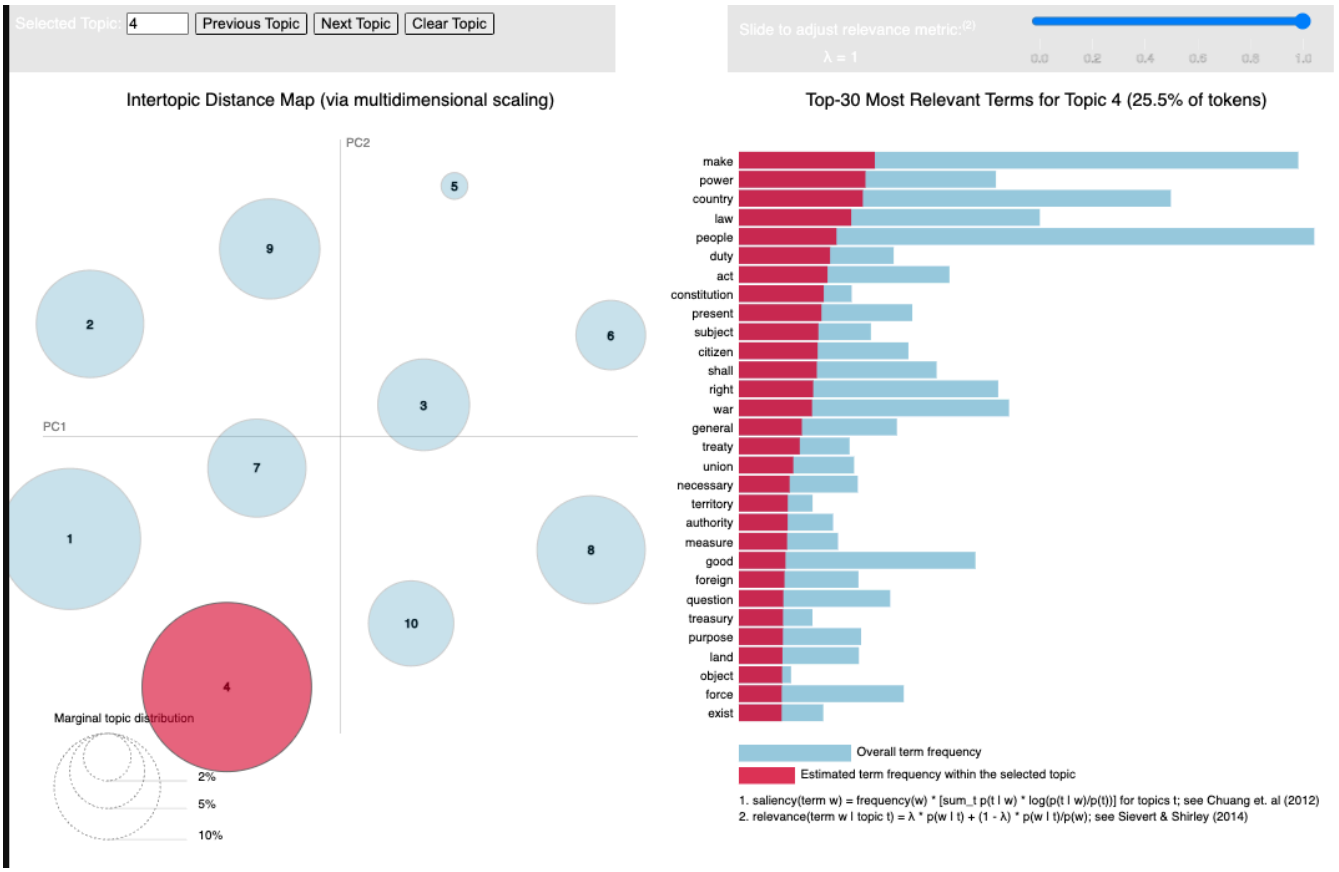
- **Distinctiveness:** We looked at how clearly the topics were separated — both in terms of their keyword overlap and visually using topic maps.

- **Comparison:** Finally, we compared how well LDA and BERTopic performed using these criteria, to see which method gave clearer and more useful topics for this dataset.

3.5 Visualization of Results

We will used visualizations to help us in interpreting and comparing the models:

1. **LDA Visualization:** The pyLDAvis library was used. The pyLDAvis.gensim_models.prepare() function was called with the trained lda_model, corpus, gensim_dict, and importantly mds='tsne' to use t-SNE for the 2D projection, aligning with the preferred visualization method for BERTopic in this case. sort_topics=False was used to maintain the original topic numbering. The interactive pyLDAvis.display() output provides:
    - An Intertopic Distance Map showing topic sizes and proximity based on Jensen-Shannon divergence projected into 2D via t-SNE.
    - Bar charts showing the top keywords of any selected topic, which can be adjustable based on relevance metric lambda (λ).



1. LDA KEYWORD BAR CHART

This shows our top 10 keywords for each topics:

2. **BERTopic Visualization:**

I made use of Bertopic library's built-in Plotly-based functions:

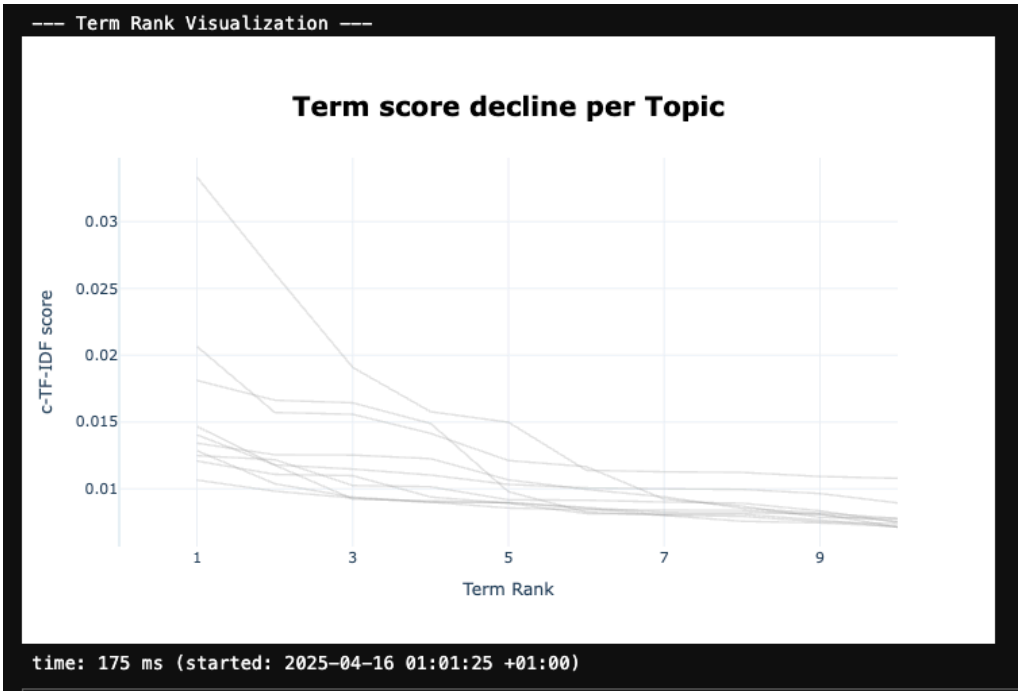
- visualize_barchart(): Displayed the top keywords (based on c-TF-IDF scores) for each generated topic.
- visualize_topics(): Generated an interactive 2D Intertopic Distance Map, using UMAP for the initial dimensionality reduction and then typically t-SNE or similar for the final 2D plot projection.
- visualize_hierarchy(): Created a dendrogram showing hierarchical relationships between topics based on cosine similarity of their c-TF-IDF vectors.
- visualize_heatmap(): Generated a heatmap showing pairwise cosine similarity scores between topic representations.

We can see Bert topics generated keywords, grouped based on 10 topics and each topic shows five keywords and the bars show their frequency in the corpus.

BERTopic Term Rank Visualization(visualize_term_rank()):

This plot is used to show the rank or score(c-TF-IDF) of selected terms across different topics. Topics are ordered in the x-axis by their actual size(number of documents). The lines you see track each individual term.

This helps us to tell the words that are important across many topics(flatter lines near the top) versus words that are specific to only a few topics(lines that spike significantly for certain topics). We can see a general downward trend and this means that words that are common across the entire corpus become less important or smaller in more specific/niche topics.



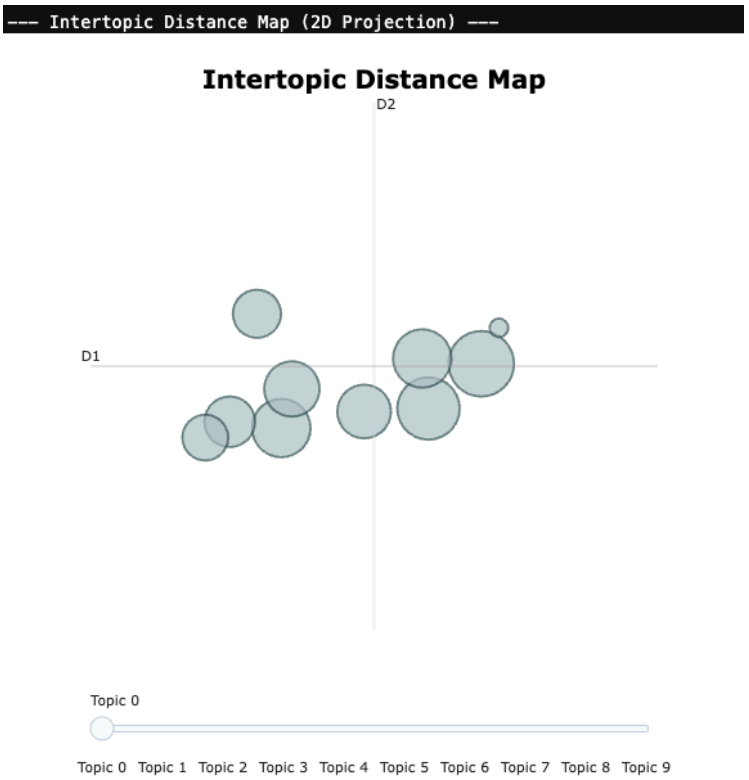
BERTopic visualize_hierarchy() Dendrogram:

This is a tree diagram (dendrogram) that shows how BERTopic topics are grouped hierarchically based on similarity. Topics that merge closer to the bottom are more similar. Branches merging higher up reps broader thematic groups.

BERTopic visualize_topics() Intertopic Distance Map:

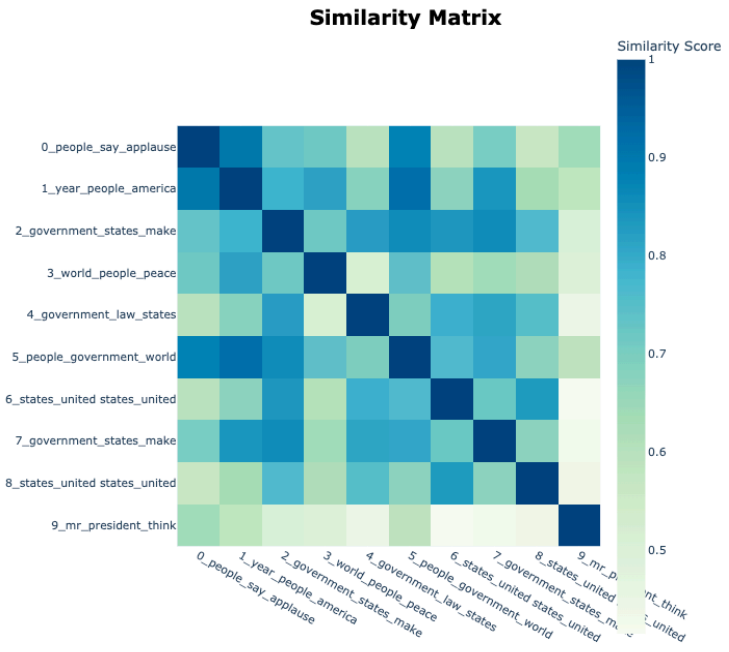
This shows us a plot of BERTopic topics in 2D based on their similarity (using UMAP/t-SNE on topic embeddings/vectors). Circle size shows topic frequency (number of documents).

From this we can see which BERTopic topics are distinct (far apart) vs. similar (close/overlapping).



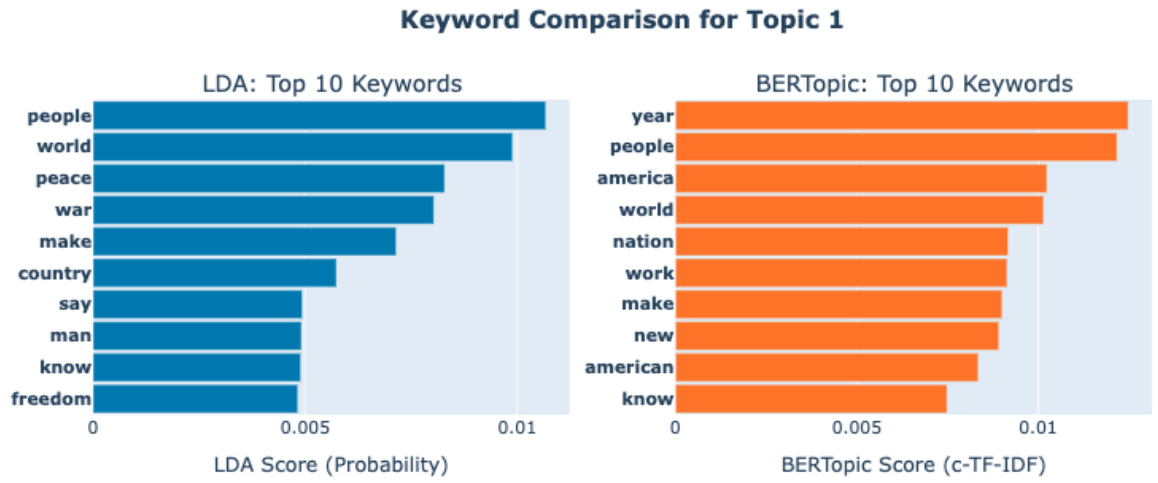
BERTopic visualize_heatmap() Heatmap

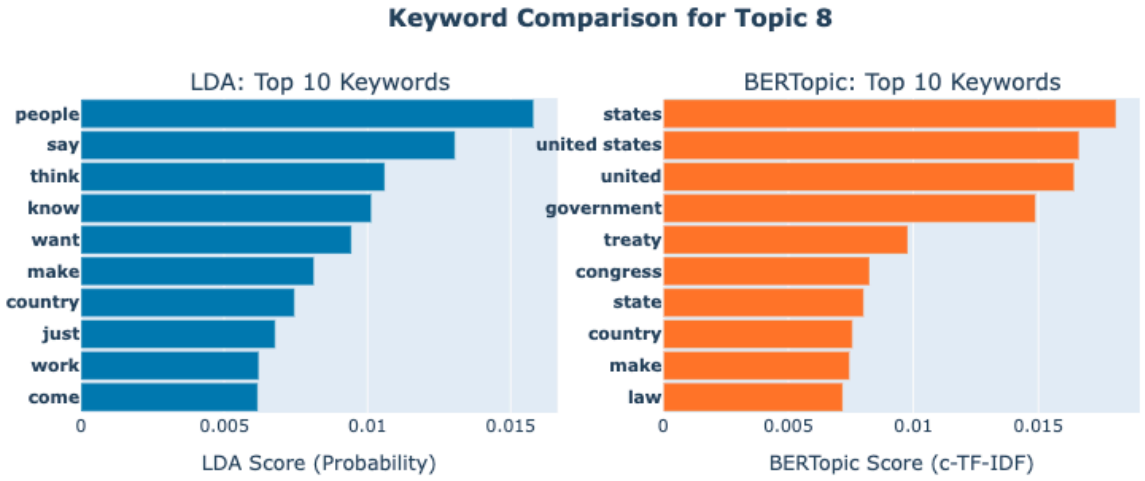
This shows us a matrix that tells us the pairwise similarity score (usually cosine similarity) between all BERTopic topics. The squares or blocks of lighter colors, shows us groups of highly similar topics. Darker colors show pairs of topics that are very dissimilar.



Uniform Keyword Charts

In order for us to get a direct comparison of topic content, I made a custom function (plot_keyword_comparison) using Plotly. This function extracted the top 10 keywords and their respective scores (probabilities for LDA, c-TF-IDF for BERTopic) for a given topic ID from both models and displayed them as side-by-side horizontal bar charts within a single figure.





RESULT ANALYSIS AND DISCUSSION

This section presents an analysis and comparison of the topics generated using LDA and BERTopic models.

4. 1. LDA Model Interpretation & Grouping

Topic ID	Top Keywords LDA	Suggested Topic Label (LDA)	Proposed Grouping (LDA)	Group Theme (LDA)
0	law, make, country, shall, service, act, department, increase, general, present	Legislation & Administration	Group A	Historical Governance & Diplomacy
1	people, world, peace, war, make, country, say, man, know, freedom	International Relations, War & Peace	Group B	20th/21st Century Foreign Policy & Conflict
2	world, peace, people, war, soviet, make, freedom, country, force, free	Cold War & International Relations	Group B	20th/21st Century Foreign Policy & Conflict
3	make, power, country, law, people, duty, act, constitution, present, subject	Governance & Constitutional Duty	Group A	Historical Governance & Diplomacy (19th C Focus)
4	tariff, country, increase, make, line, duty, rate, article, schedule, product	Trade & Tariffs	Group C	Economic & Domestic Policy
5	people, federal, country, make, world, economic, national, shall, business	National Economy & Federal Role	Group C	Economic & Domestic Policy
6	man, people, make, think, war, world, right, say, country, come	General Discourse & Human Condition	Group D	Political Discourse & Foundational Ideas
7	applause, people, make, work, know, job, country, americans, just, say	Domestic Issues, Economy & Jobs	Group C	Economic & Domestic Policy
8	people, say, think, know, want, make, country, just, work, come	Public Opinion & Discourse	Group D	Political Discourse & Foundational Ideas
9	people, tax, make, program, increase, country, work	Fiscal Policy & Taxation	Group C	Economic & Domestic Policy

Interpretation:

- LDA was able to successfully identified several distinct and interpretable themes: Cold War (Topic 2), Trade/Tariffs (Topic 4), Fiscal Policy/Taxation (Topic 9), and broader themes of Governance (Topics 0, 3, 5) and International Relations (Topic 1).

- Topics 6 and 8 seem more generic, capturing common words used in public discourse ("people," "make," "say," "think," "country") and might represent a more general "speech-making" topic.
- The presence of "applause" in Topic 7 is interesting, potentially indicating topics discussed in speeches receiving immediate audience feedback.

4.2. BERTopic Model Interpretation & Grouping

Topic ID	Top Keywords BERTopic	Suggested Topic Label (BERTopic)	Proposed Grouping (BERTopic)	Group Theme (BERTopic)
0	people, say, applause, know, president, america, make, want, year, thank	Public Address & Domestic Focus	Group D	Political Discourse & Communication
1	year, people, america, world, nation, work, make, new, american, know	State of the Nation / Time	Group E	Broad National & World Themes
2	government, states, make, united, united states, public, congress, country, law	US Governance & Federal Structure	Group F	US Governance & Law
3	world, people, peace, war, president, united, nation, make, force, year	International Relations, War & Peace	Group G	Foreign Policy & Conflict
4	government, law, states, make, congress, united states, united, power, state, country	Governance & Lawmaking	Group F	US Governance & Law
5	people, government, world, nation, man, make, year, great, american, time	General Discourse & Nationhood	Group E	Broad National & World Themes
6	states, united states, united, government, state, law, shall, make, congress	Constitutional Law & Federalism	Group F	US Governance & Law
7	government, states, make, congress, year, country, united states, united, great	Governance & Public Affairs	Group F	US Governance & Law
8	states, united states, united, government, treaty, congress, state, country, make	Treaties & Federal Relations	Group F /Group G	US Governance & Law / Foreign Policy
9	mr, president, think, say, people, question, year, senator, governor, make	Political Dialogue & Process	Group D	Political Discourse & Communication

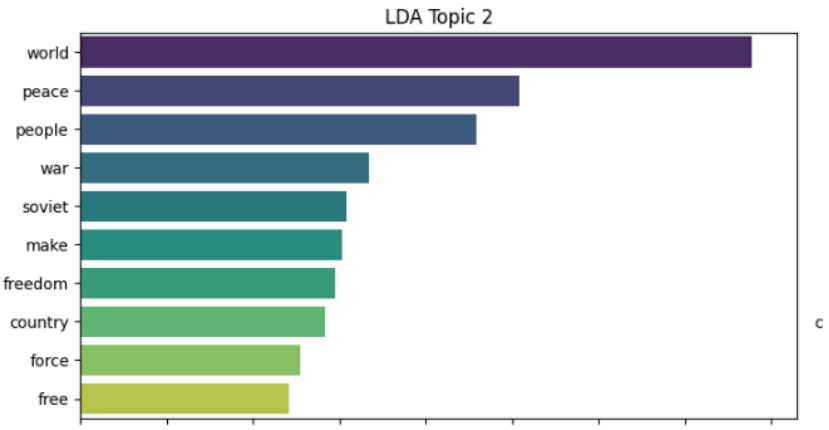
Interpretation:

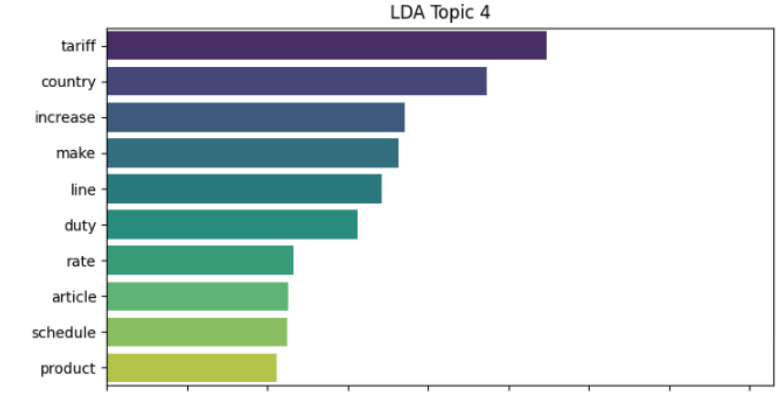
- BERTopic was able to identify core themes like Governance (Topics 2, 4, 6, 7 - potentially capturing different facets), International Relations (Topic 3), and Public Address (Topic 0).
- It surfaces slightly different nuances: a topic seemingly related to the "State of the Nation/Union" address structure (Topic 1), a distinct topic on Treaties (Topic 8), and one potentially reflecting interactive formats or addressing specific roles (Topic 9).
- Semantic grouping seems evident; for example, multiple topics relate to "government," "states," and "law," but potentially capture different contexts (general governance vs. constitutional law vs. treaties).
- Topic 5 appears somewhat general, similar to LDA's general discourse topics.

4.3. Comparative Analysis: LDA vs. BERTopic:

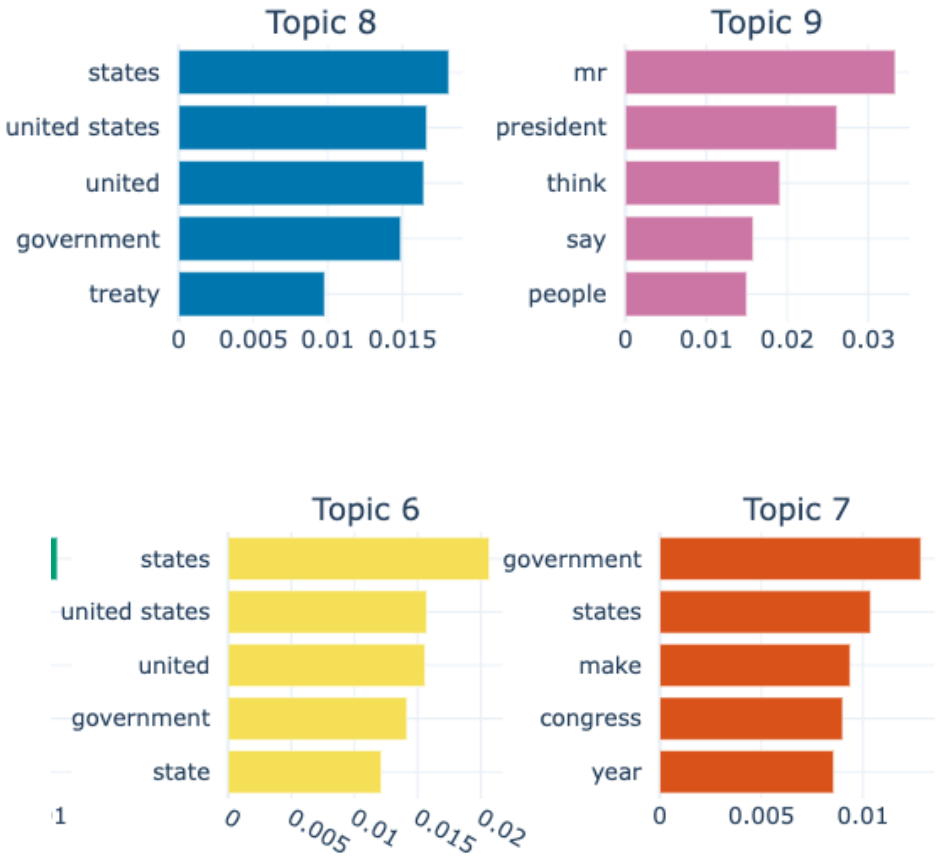
A comparison of the results from the two models reveals interesting similarities and differences:

1. **Similar Themes:** Both models were able to identify core themes like Governance/Law, International Relations/War & Peace, and General Public Discourse.
2. **LDA Strengths:** LDA did a better job at picking out very specific topics based on how keywords appear from our results , such as Trade/Tariffs (LDA 4) and Fiscal Policy/Tax (LDA 9). Even the Cold War theme showed the keyword "soviet" (LDA 2).





3. **BERTopic Strengths:** BERTopic showed its effectiveness at grouping documents based on broader semantic meaning, leading to richer keyword generation which showed the benefits of c-TF-IDF and inclusion of n-grams(1,3), but had some overlapping in governance topics (like 2, 4, 6, 7, 8). Bertopic was also able to clear themes like Treaties (BERTopic 8) and Political Dialogue (BERTopic 9) that LDA didn't highlight as well. Its topics sometimes felt more "modern" and context-aware thanks to the transformer embeddings (though this depends heavily on the dataset's time range).



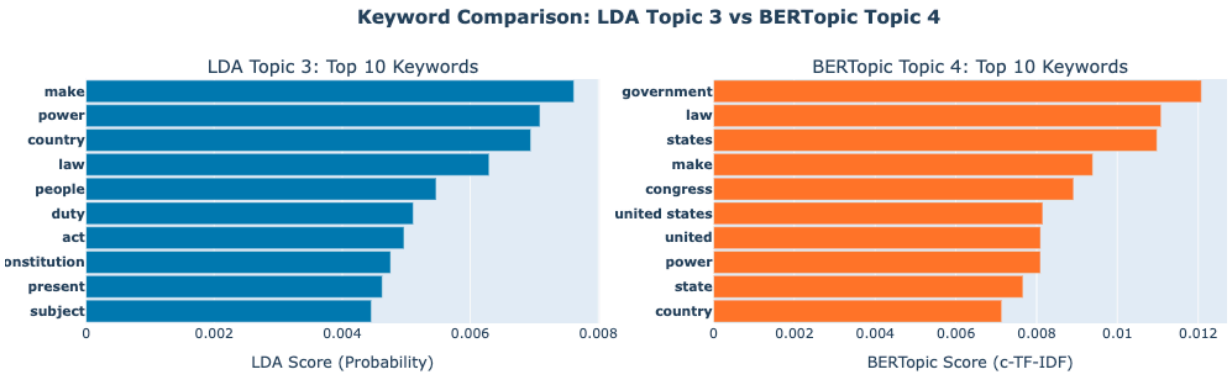
4. **Differences in Keywords:** The keywords generated by LDA keywords are based on probability distributions, while BERTopic's are based on c-TF-IDF scores, measuring word importance within a topic cluster versus the total word frequency across the entire corpus. This leads to slightly different keyword lists even for similar themes.
5. **Topic Granularity:** BERTopic, has the potential and capability if we use a different clustering algorithm (like its default HDBSCAN), to give a totally different number of topics and keywords. Using K Means with K=10 made it easier for me to compare with LDA, but this might not be so with another clustering algorithm which might have generated more or less themes.
6. **Interpretability:** Both models were able to produce good, understandable and interpretable topics, although both LDA & BertTopic had 1-2 topics that were more generic.

4.4 Keyword Comparison Based on Same/Similar Topic

Okay, let's compare two examples of similar topics identified by LDA and BERTopic, focusing on the keyword differences and what they might tell us about how each model works.

Example 1: Topic related to Governance and Law

- **LDA Topic 3:** make, power, country, law, people, duty, act, constitution, present, subject
- **BERTopic Topic 4:** government, law, states, make, congress, united states, united, power, state, country

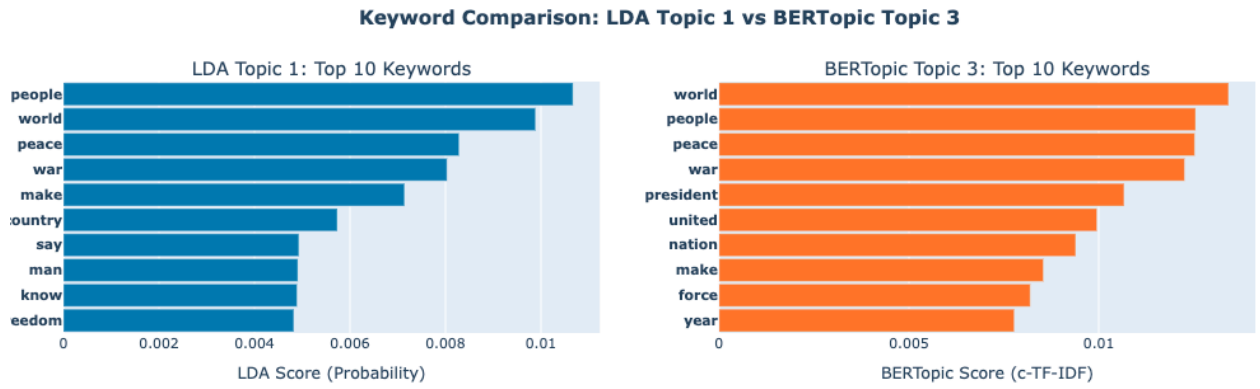


Comparison:

1. **Shared Core:** Both topics showed that it revolved around governance, sharing keywords like make, power, country, and law.
2. **LDA Focus:** LDA includes people, duty, act, constitution, present, subject. These keywords show the governance principle and responsibility centric theme. Words like ‘constitution’ & ‘duty’ give more depth. LDA identifies these potentially because these specific words frequently *co-occur* in documents discussing these principles.
3. **BERTopic Focus:** BERTopic produced keywords like government, states, congress, united states. These are concrete *keywords when it comes to governance in the* US political system. Thanks to its use of semantic embeddings, it grouped documents that discuss governance structures even when the wordings were slightly different. It was able to pick up the meaning of terms in context of its application in the corpus,
4. **Interpretation of Difference:** LDA was able to pick up abstract ideas and terms like("constitution," "duty") by looking at hie words that appeared together. BERTopic on the other hand was able to focus more on concrete government structures like "states," "congress," "government") based on semantic similarity.

Example 2: Topic related to International Relations / War & Peace

- **LDA Topic 1:** people, world, peace, war, make, country, say, man, know, freedom
- **BERTopic Topic 3:** world, people, peace, war, president, united, nation, make, force, year



Comparison:

1. **Shared Core:** The central theme for both models was international relations, conflict, and diplomacy, with some shared keywords like “people, world, peace, war, and make.”
2. **LDA Focus:** LDA had keywords like country, say, man, know, freedom. While country and *freedom* fit well, especially since *freedom* often appears alongside *war* and *peace* in presidential speeches, others like *say*, *man*, and *know* are more general and likely reflect common speech patterns in those documents.
3. **BERTopic Focus:** BERTopic was able to include concrete & specific terms like president, united, nation, force, year. These are more specific to the *actors* ("president"), "United Nations"), *actions* ("force"), and timing ("year"). The semantic model connects documents discussing concrete actions and entities involved in global affairs.
4. **Interpretation of Difference:** LDA was able to capture common co-occurring words like *freedom*, though some are less topic-specific. BERTopic was able to capture and identify key players, actions, and context — giving a better and clear keyword pattern & meaning rather than just word patterns.

Finally, we are able to see and learn the following after applying both models to the same corpus:

1. **Dominant Themes:** Looking at both models, the major themes we had in US presidential speeches were around themes like Governance (domestic policy, law, constitution), International Relations (war, peace, foreign nations), the Economy (trade, fiscal policy, jobs), and the act of Public Address itself (addressing the people, the nation)
2. **Model Differences:** we can see the major difference between LDA's probabilistic word co-occurrence vs. BERTopic's semantic embedding and clustering and how they both clearly gave different topic nuances.
3. LDA thrived at finding topics defined by specific, frequently co-occurring keywords, while BERTopic was excellent at grouping documents discussing similar concepts, even if they use slightly different wording.
4. **Overlapping Topics:** Some overlap between topics is expected and natural in real-world text. Governance, economy, and foreign policy are often intertwined in presidential discourse. Both models showed some topics with overlapping keywords (e.g., LDA 6 & 8; BERTopic 2, 4, 6, 7).

4.5. Performance Comparison: Cloud vs. Local

For this project, I performed this topic modeling experiments on both Google Colab (cloud) and a local Linux environment (Macbook Air M1 via Anaconda via Jupyter Notebook) to compare performance.

This table contains a detailed analysis and comparison of both cloud and local performance:

Category	Cloud (Google Colab)	Local (Your Machine)
Environment	Google Colab Standard Runtime	Macbook Air M1 via Anaconda via Jupyter Notebook
Hardware	T4 GPU	GPU: M1 Chip
Preprocessing time	10 mins 46 secs	19 mins 26 secs
LDA Training Time	2 mins 51 sec	2 mins 32 sec
BERTopic Training Time	1 mins 8 secs	1 mins 9 sec
LDA Performance Summary	Similar to local local]	Slightly faster but almost similar to Colab]
BERTopic Performance	Slightly faster due to GPU	Just a second difference
Preprocessing/Visualization	Fast with negligible differences	It was way slower compared to using a GPU
Conclusion	Excellent for DL models like BERTopic with GPU acceleration	Suitable for LDA and lightweight tasks, limited for DL models

CONCLUSION AND RECOMMENDATION

5. 1. Conclusion

Now for this project, I was able to conduct a comparative analysis for two topic modelling methods which are, LDA and BERTopic.. The aim was to discover topics, thematic structures from a manually collected corpus of 1061 U.S. Presidential speeches. The primary research question was to compare both algorithms and I can confirm that they proved capable of identifying distinct and broadly interpretable topics, though they yielded different thematic perspectives.

For LDA, I used a heavily preprocessed text and a probabilistic Bag-of-Words approach. This was the core and it helped in identifying clear, policy-focused topics like Tariffs/Trade, Fiscal Policy, and Cold War themes. Its strength lies in spotting frequently co-occurring terms, which made the topics coherent but sometimes broad.

For BERTopic, working with the same preprocessed text but leveraging Bert's contextual embeddings, KMeans clustering (with 10 forced topics), and c-TF-IDF with n-grams, I picked up different nuances. It captured communication styles (like Public Address, Political Dialogue) and had a strong focus on U.S. governance mechanics. While preprocessing may have limited it somewhat (compared to using raw text), its use of semantic context and n-grams led to richer keyword groupings in some areas.

In terms of Model Comparison, Neither model was clearly better overall. LDA gave more distinct topics around specific policy areas, while BERTopic offered deeper insights into discourse style and structural elements of governance. BERTopic also stood out with built-in visualizations like hierarchies and heatmaps. The choice depends on the analytical goal.

In conclusion, This task and report showed that both traditional (LDA) and modern (BERTopic) topic modeling methods can work well on this historical dataset. A key factor in success was the preprocessing pipeline — especially the use of custom, domain-specific stopwords — which helped LDA and gave BERTopic a solid baseline to build from.

5.2. Recommendation and Future Work:

My report shows both the strengths and trade-offs of both LDA and BERTopic, but there's still a lot of room to explore and improve how topic modeling is applied to a corpus of similar manner.

1. Rethink my Preprocessing approach for BERTopic

BERTopic might benefit from a less preprocessed input. Since it relies on semantic embeddings rather than raw word frequency, stripping the text down too much might actually remove useful context. Future experiments should try using minimally processed text , perhaps just lowercasing and basic punctuation cleanup and compare the results to what we've already seen with the heavily preprocessed version. This could give us a clearer sense of how much preprocessing helps or hurts topic quality with BERTopic.

2. Letting BERTopic's default clustering method (HDBSCAN) generate its own number of topics.
For a fair comparison, both models I had to set Bertopic to generate 10 topics. But this number might not be ideal. In follow-up work, it would be worth:

- Letting BERTopic's default clustering method (HDBSCAN) generate its own number of topics by itself and compare its output to KMeans.

3. Use Metadata More Intentionally

The speeches in this dataset come with rich metadata — names, dates, political context — that weren't really used in this analysis. That's a missed opportunity. Future work could tap into this.

4. Better Topic Interpretation

While word lists are helpful, they don't always capture the full picture. BERTopic offers features like sample documents or key sentences for each topic — these could make interpretation much easier and more grounded. It's also worth tweaking how topic keywords are generated (via c-TF-IDF) to see if the results can be improved by adjusting for things like document length.

5.3. Ethical, Legal, and Professional Considerations

Here are some ethical considerations to note from this task:

- Source Data Bias: For a manually collected dataset, with different topics, themes, different range of year, it may seem too specific and directed towards one major topic, so this should be noted.
- Subjectivity of Interpretation: The topics and themes generated are subjective to different interpretations, it is not a one shoe fit all thing.
- Potential for Misuse: All political analysis done can always potentially be misused so it is important to state that this project focuses on the technological implications and difference and not on any real world issue or a basis for any propaganda.
- Data Provenance and Accuracy: I acknowledge that this system is not perfect and might contain eros, so if these would be used in a real world scenario, humans should also be able to cross check results that this model produces

REFERENCE

1. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*(Jan), 993–1022.
2. The American Presidency Project. (n.d.). University of California, Santa Barbara. Retrieved [April 16, 2025], from https://www.presidency.ucsb.edu/
3. University of Virginia Miller Center. (n.d.). Retrieved [April 16, 2025], from https://millercenter.org/the-presidency/presidential-speeches
4. Grootendorst, M. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*. https://arxiv.org/abs/2203.05794/
5. McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv:1802.03426*. https://aclanthology.org/D19-1410/
6. Dr. Ali Almeer's GitHub**:** Almeer, A. (n.d.). *Ali Almeer – GitHub profile*. GitHub. https://github.com/Ali-Alameer
7. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. Proceedings *of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Association for Computational Linguistics. https://aclanthology.org/D19-1410/
8. Sievert, C., & Shirley, K. (2014). LDAvis: A method for visualizing and interpreting topics. *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, 63–70. Association for Computational Linguistics. https://aclanthology.org/W14-3110/