COMPSCI 4NL3 Homework 2: Corpus Analysis

Alan Zhou

February 2025

1. **Dataset**

   The dataset chosen for this Homework comes from the Congressional hearings source. In particular, I chose congressional hearings all related to Donald Trump. The two categories chosen are: hearings shortly after he won the election in 2016, and hearings shortly after he lost the election in 2020. Each document is a separate hearing, meaning that I collected a total of 200 hearings to complete my dataset.

   I chose this dataset since I have a passion for at least watching the drama that comes with elections, let alone the fact Donald Trump is one of the most polarizing candidates for the presidency. I thought that it could be interesting to see if there are any specific words, or words that can be tied to a specific event or emotion, based on the category.

2. **Methodology**

   (a) Combining Text Files
   The script combines all of the .txt files containing the congressional hearings from two folders: cong_recs/lose and cong_recs/win into two separate output files: loset.txt and wint.txt, respectively.

   (b) Text Preprocessing
   The script preprocesses the text with the same methodology used in homework 1: Counting Tokens, including the following filters: lowercasing, removing punctuation, removing stopwords, stemming and lemmatization. The only difference is in addition to the custom filter to remove words less than 3 characters long, I also included the option to remove words that I guess to be giveaways, or not that insightful. ChatGPT was used here just like in homework 1.

   (c) Preprocessing Files
   After the script reads the combined files, and preprocesses the text, the processed tokens are saved to new files: processed_loset.txt and processed_wint.txt.

   (d) Creating Dictionary and Corpus
   Using the gensim library, the script creates a dictionary and corpus. Each word is mapped to a unique ID, and the corpus represents the documents as bags of words.
   This is necessary for topic modeling, as the formatting can now be processed by the LDA model.

   (e) LDA Topic Modeling
   The script runs Latent Dirichlet Allocation (LDA) topic modeling using the gensim library. It specifies the two topics (aligning with "win" and "lose"), and runs the model with a different amount of passes. Deepseek was used a little bit for guidance and debugging in this section as well as in Visualization.

   (f) Visualization
   Using pyLDAvis, a visualization tool for LDA models, the script produces a visual in a created HTML

file: (lda_visualization.html). This allows for easier interpretation by users to understand the distribution of words within each topic.

(g) Results
The script displays the top 25 words for each topic, and they are saved to a created file: lda_topics.csv. These words are also printed to the console.

(h) Naive Bayes
Next, the script prepares the labeled data for Naive Bayes by associating the preprocessed tokens with their respective labels: win or lose. This labeling is required in order to further analyze through Naive Bayes.

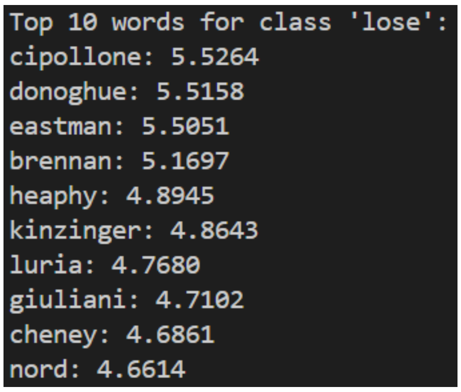
(i) Counting
Finally, using Naive Bayes, the script counts every instance of each word in the labeled data and computes the log-likelihood for each word. This allows for users to identify words that are the most indicative, or suggesting of each class: win or lose.
The script also prints the top 10 words for each class: win and lose to the console, based on log-likelihood scores.

Libraries used:

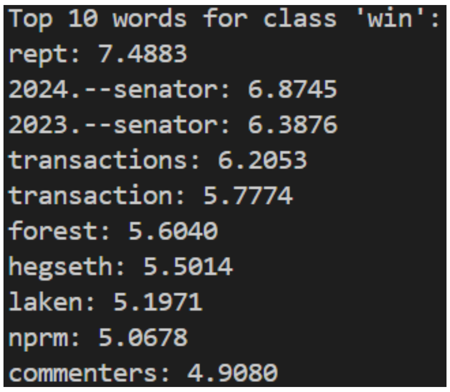
1. nltk: for stemming and lemmatization (same as homework 1: Counting Tokens)
2. gensim: for LDA topic modeling and creating dictionary and corpus
3. pyLDAvis: for visualization LDA results
4. pandas: for displaying topic data in table

3. **Results and Analysis**

Log-likelihood Ratio (2.3):



```
Top 10 words for class 'lose':
cipollone: 5.5264
donoghue: 5.5158
eastman: 5.5051
brennan: 5.1697
heaphy: 4.8945
kinzinger: 4.8643
luria: 4.7680
giuliani: 4.7102
cheney: 4.6861
nord: 4.6614
```



```
Top 10 words for class 'win':
rept: 7.4883
2024.--senator: 6.8745
2023.--senator: 6.3876
transactions: 6.2053
transaction: 5.7774
forest: 5.6040
hegseth: 5.5014
laken: 5.1971
nprm: 5.0678
commenters: 4.9080
```

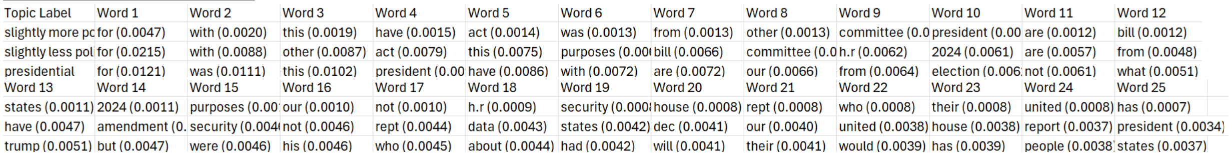
The function compute_log_likelihood calculates the log-likelihood ratio for each word in the vocabulary, given the two classes: win and lose. This ratio measures how much more likely a word is to appear in a specific class compared to other classes.

The alpha parameter activates add-one smoothing, to ensure that no word has a zero probability. Above is the corresponding output for the congressional hearings.

There are a couple interesting conclusions that I drew after seeing these lists. The most distinguishing word associated with the class 'lose' happens to be cipollone. I did not know what this word was referring to, but after one quick search, I discovered that Pat Cipollone was Donald Trump's attorney, defending him during his first impeachment trial, and played a key role in the January 6 committee hearings. This makes perfect sense since one could imagine that an impeachment would be a strong argument to not re-elect said president, let alone what happened to the Capitol on Jan 6th 2021. Looking at top 10 words for 'win', I instantly noticed hegseth. Now being older and more caught up with news, I recognized that this was Pete Hegseth, Donald Trump's nominee for Secretary of Defense. Being number seven on the list seems consistent with real world news, since I am aware there is controversy surrounding him, and his past habits, questioning his suitability for the role.

The main takeaway after seeing these lists is the fact that most of these "words" are actually names of people. To improve the model, I would suggest implementing a normalization option designed to remove any names of people. That way, there might be words related to events or topics, rather than significant people politically tied to Donald Trump, providing a different insight into his presidency.

Topic Modeling (2.4):



Topic Label	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10	Word 11	Word 12
slightly more pc	for (0.0047)	with (0.0020)	this (0.0019)	have (0.0015)	act (0.0014)	was (0.0013)	from (0.0013)	other (0.0013)	committee (0.0	president (0.00	are (0.0012)	bill (0.0012)
slightly less pol	for (0.0215)	with (0.0088)	other (0.0087)	act (0.0079)	this (0.0075)	purposes (0.00	bill (0.0066)	committee (0.0	h.r (0.0062)	2024 (0.0061)	are (0.0057)	from (0.0048)
presidential	for (0.0121)	was (0.0111)	this (0.0102)	president (0.00	have (0.0086)	with (0.0072)	are (0.0072)	our (0.0066)	from (0.0064)	election (0.006	not (0.0061)	what (0.0051)

Word 13	Word 14	Word 15	Word 16	Word 17	Word 18	Word 19	Word 20	Word 21	Word 22	Word 23	Word 24	Word 25
states (0.0011)	2024 (0.0011)	purposes (0.00	our (0.0010)	not (0.0010)	h.r (0.0009)	security (0.000	house (0.0008)	rept (0.0008)	who (0.0008)	their (0.0008)	united (0.0008)	has (0.0007)
have (0.0047)	amendment (0.	security (0.004	not (0.0046)	rept (0.0044)	data (0.0043)	states (0.0042)	dec (0.0041)	our (0.0040)	united (0.0038)	house (0.0038)	report (0.0037)	president (0.0034)
trump (0.0051)	but (0.0047)	were (0.0046)	his (0.0046)	who (0.0045)	about (0.0044)	had (0.0042)	will (0.0041)	their (0.0041)	would (0.0039)	has (0.0039)	people (0.0038	states (0.0037)

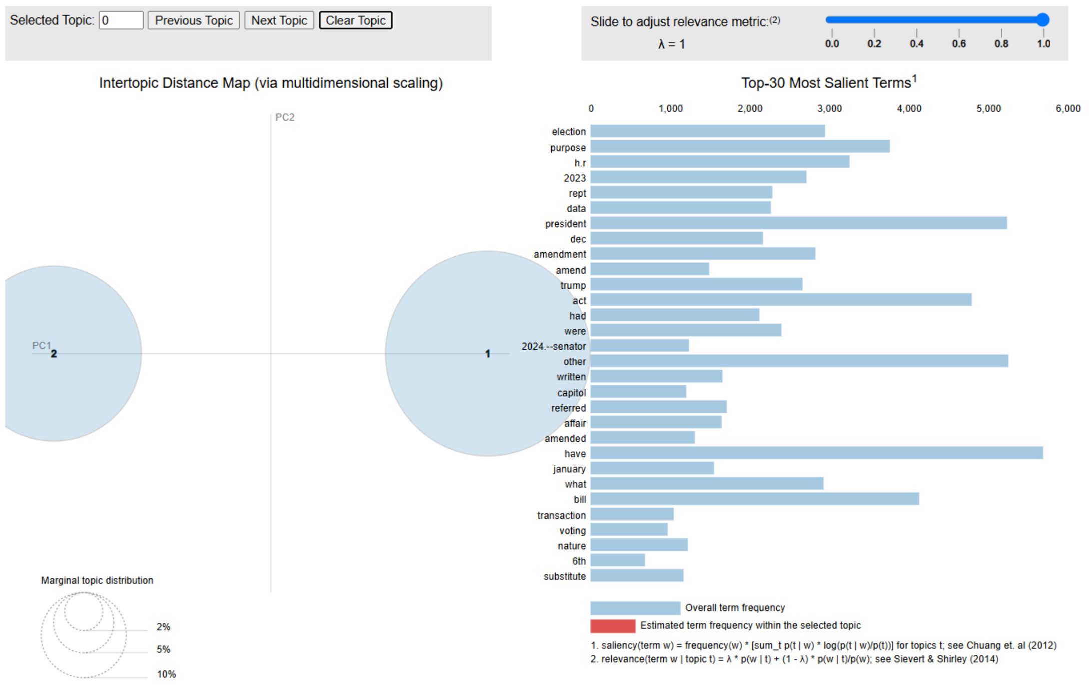
Above is the output table, with manually assigned topics: slightly more political, slightly less political,and presidential. These topics were manually chosen by me, and contain the top 25 terms for each topic, sorted by their probability of belonging to that topic, along with their probabilities.

I decided to label these topics as I did due to these specific words being present:

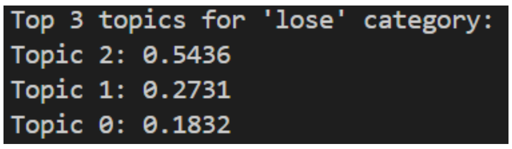
- slightly more political: states, act, committee, bill, election
- slightly less political: purposes, h.r, security, data
- presidential: president, election, people

However, I would say that these probability values associated are not very strong, and thus I would not safely label the topics based on these words.

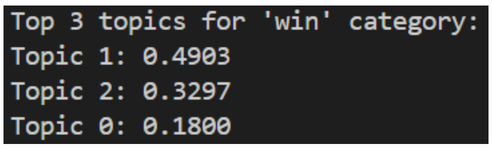
The html file is shown below:



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)



```
Top 3 topics for 'lose' category:
Topic 2: 0.5436
Topic 1: 0.2731
Topic 0: 0.1832
```

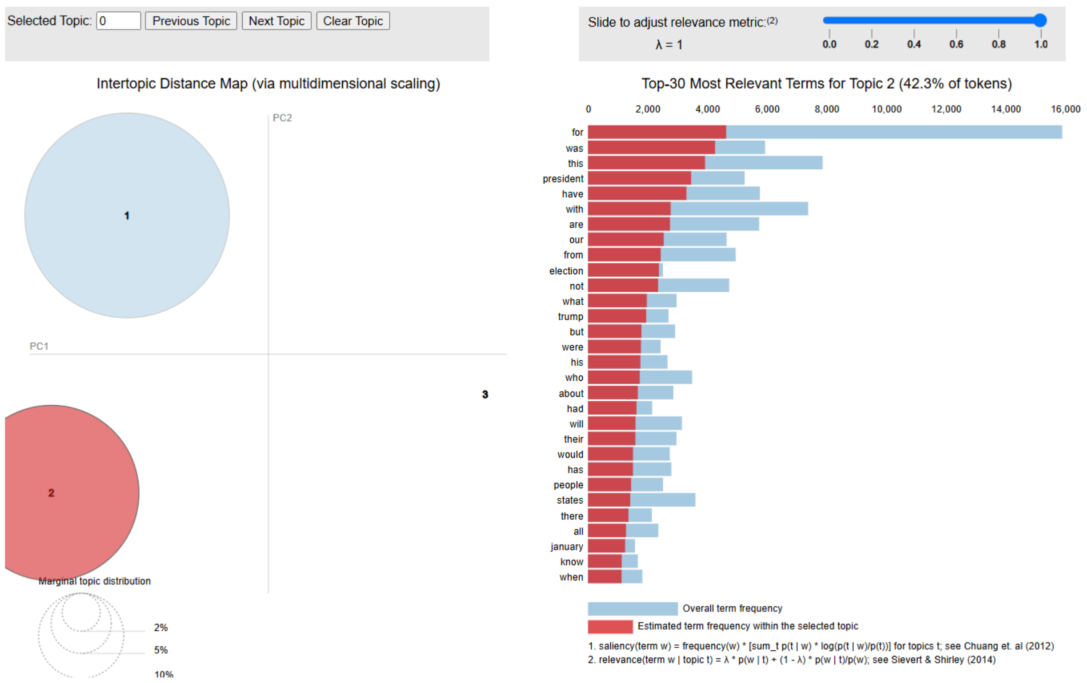


```
Top 3 topics for 'win' category:
Topic 1: 0.4903
Topic 2: 0.3297
Topic 0: 0.1800
```

Above are the average distribution of all topics, for documents in their respective categories. Topic 0, 1, 2, refers to slightly more political, slightly less political, and presidential respectively.

<u>Experimentation (2.5):</u>

The following demonstrates the most insightful result out of the different experimentation techniques.



Selected Topic: 0
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1.0
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
Marginal topic distribution
2%
5%
10%
Top-30 Most Relevant Terms for Topic 2 (42.3% of tokens)
0
2,000
4,000
6,000
8,000
10,000
12,000
14,000
16,000
for
was
this
president
have
with
are
our
from
election
not
what
trump
but
were
his
who
about
had
will
their
would
has
people
states
there
all
january
know
when
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Above is the html lda visualization when using a count variation for the bag of words. I decided to evaluate this specifically, because I instantly drew a connection from real-world events to what I produced on the right hand side. The appearance of 'january' is significant, not only because of the inauguration, but once again with reference to the events of Jan 6th 2021, a critical event, that happened towards the end of Trump's first term, after losing his re-eleciton.

Thus, even though January is the time for a new president's inauguration, with respect to this dataset, this is not the case. January is heavily correlated to the category of Donald Trump losing the election.

4. Discussion

<u>Subsection 1</u>

There are many aspects learned about this dataset after all the preprocessing and modeling. The most important conclusion to draw after this experiment is that factoring in names into the bag of words is not fair. This is because names seem to be the most distinguishable when comparing between the classes. As previously stated, it would be ideal to implement a method to remove names, then re-evaluating the log-likelihood ratio as well as the topic modeling. From our information gathered, it is nearly impossible to distinguish which words can distinctly belong to one class versus the other. This could have been foreseen since they are still all surrounding the controversies and history related to one man, within a time frame of less than a decade.

Describing, this code is analyzing a collection of texts to identify the most important words and topics within them. In this case, there is a set of documents, congressional hearings, categorized based on whether they describe Donald Trump after winning the election or Donald Trump after losing the election. The code processes these texts to clean them up, remove common words, and identify key terms that distinguish the two categories.

An important step is topic modeling, which tries to automatically find patterns in the words used in these texts. This helps us understand what kinds of themes show up more in ”win” documents versus ”lose” documents. The results show which words are most strongly associated with each category and what topics are most common in the overall dataset.

This project helps reveal the language differences in winning and losing contexts, with respect to the 2020 and 2024 presidential elections.

Subsection 2

I believe that I made a mistake when choosing my dataset. I have a passion and curiousity towards the subject, but I do not have the foundation or existing understanding of the types of contents within congressional hearings. To have a better established understanding of congressional hearings may have lead me to choose a more meaningful distinguishable dataset belonging to two classes. This would be in hopes of producing more interpretable results.

Preprocessing was very similar to that done from homework 1: Counting Tokens.

The preprocessing steps significantly impacted the quality of topic modeling. Tokenization and lemmatization improved the accuracy of word representations, allowing for better topic coherence. Stopword removal reduced redundancy, ensuring that more meaningful words contributed to topic formation.

Topic modeling was conducted using Latent Dirichlet Allocation (LDA). The resulting topics were coherent and interpretable, aligning well with expected themes within the dataset. Example topics included political themes, each characterized by distinctive sets of keywords that portrayed that theme.

One challenge was the sensitivity of LDA to parameter tuning. The choice of topic number significantly influenced results, requiring iterative refinement to avoid overfitting or underfitting topics.

Overall, our approach to data processing and topic modeling could have yielded more insightful topic distributions. The preprocessing pipeline effectively enhanced downstream analysis, although certain information loss and sensitivity to parameter tuning, remain considerations for future improvement. By addressing these limitations, further improvements can be made to optimize the extraction of meaningful themes from textual data.