CCROWN and DBEERS
Advanced Analytical Tools – Prof. Rayguru
Final Project Report

Introduction

In January of 2025, President Donald Trump was elected into office for his second term. For those familiar with his previous two campaigns, this most recent cycle had a very different approach and we saw a much more calm and rehearsed Trump compared to 2016 and 2020. Although half of 2024's election was against Kamala Harris, it is still both germane and interesting to look back at 2020 to see how Trump debated against Joe Biden the first time. In this vein, we created a dataset that concatenated every sentence both Trump and Biden said during their first presidential debate on Sept 29th of 2020 in Cleveland, Ohio.

Data Preprocessing

All of our initial data processing produced two contiguous corpora of everything that Biden said during the debate and everything Trump said. This was done by eliminating all rows where the moderator, Chris Wallace, was identified as speaking and then combining all of Biden and Trump's sentences into two large, separate, lowercase units. We made sure to do all of the following steps one at a time because many of our different models and analyses required different versions of these corpora. Some required removing nltk's standard stopwords and some additional ones of our own* while others needed tokenized into just sentences… or sentences AND words… or just words.

*Ex: the term "crosstalk" was added through the audio extraction technique anytime the two candidates were speaking over each other. If you're familiar with this debate, it was before the auto-time-shutoff mic's that were used in 2024 and this happened often. So "crosstalk" was one of several stopwords that nltk did not catch but that we removed, manually in a list.

EDA and Insights from Word Frequency

Because this was such a small corpus (only 95 minutes) and the two candidates are so different, many of our insights came from word frequency analysis and simple EDA.

- Trump spoke about 12% more than Biden did
- Biden had a slightly higher vocabulary but said more stopwords
- Biden's sentences, on average, were longer than Trump's

Tokenizing by word and then using Counter, from Collections, we were able to identify the top 60 most frequently said words (ranging from 73 - 7 and 66 – 9 total occurrences for Biden and Trump, respectively) for each candidate (Figures 1a and 1b).

- Biden's top 10 words focused on numbers, facts, votes and the presidency
- Trumps top 10 words were heavily focused on Biden, the country or were action verbs
- America, American, United, or States were not in Trump's Top 60 words

We then plotted a paired histogram for the Top 60 words spoken by both candidates – these were terms both Biden and Trump said frequently (Figure 2). This was interesting (Trump said almost every shared word more times than Biden) but led us to a much better comparison: which Top 60 words were unique to each candidate? Side by side histograms (Figure 3) of their distinct Top 60 words gleaned many more insights:

- Biden talked about the courts and COVID while Trump did not
- Biden talked about the economy and healthcare while Trump talked about the country and "Obamacare"
- Biden talked about jobs, taxes and votes while Trump stressed China and the military
- Biden stressed facts and the truth (via "true"), numbers and plans while Trump talked about Biden (via "Joe")
- Overall, and after a manual part-of-speech adjustment, Biden said more topic-based nouns(17) than Trump (11) – who used mostly verbs, names and connecting words

N-Gram Models

Moving forward, we wanted to see what patterns we could discern for each candidate. We did this using Bigram and Trigram calculations*. These models calculate conditional probability of words based on frequency of word pairs/triplets. These analyses had two parts which used two different corpora. For identifying common bigram/trigram pairs we used a stopword-removed corpus. This was so we could look for meaningful word combinations (although we did decide to conduct one bigram analysis with stopwords that led to some very interesting results). But for building ngram language models using MLE (Maximum Likelihood Estimation – a conditional probability approximator using counts) we included stopwords so that we could preserve context and use these tools later, for text generation (Figures 4-6). Using bigram models we found insights from each of their top 20 most frequent word pairings:

- Biden talks a lot about the American people and the United States, listing plans and arguments by number, affordable care, taking care, making things clear, what people want, the outcome of the election, the facts and the Democratic party

- Whereas Trump frequently (4 out of Top 20) uses large numbers to demonstrate his points, talks about law enforcement and law and order, mocks Biden with "oh really", and speaks about negative things during the “radical left’s” time in office calling them to “look (at what) happened” and “people died”
- While both candidates have “people” as their top term in their word frequency model and Biden says it more total times than Trump, Trump has more (4 out of Top 20) top bigrams containing the term “people” than Biden
- Using bigram frequency with stopwords:
    - Biden’s bigrams still primarily concern the American people, the US and the facts
    - Where Trump’s, predominantly, change to person-centric statements such as “I think”, “I want”, “I know”, “I tell”, etc.

Using a trigram model, we can see even more clearly. This model adds more context (via surrounding words) and their Top 20 most frequent word triplets almost unanimously reinforce our findings from the bigram model. Beyond simple topic keywords, these models help us learn more about how these candidates are approaching the debate. Biden talks more about the issues (nouns), what the American people want, his plans for the future and trying to establish facts, while Trump talks about himself, attacks his opponent and tries to appeal to the masses (especially those with simpler educations) by referencing them more often, using simple, action-based English and quoting large numbers and bad outcomes to create senses of importance and urgency.

*Unigram calculations are present in the code but are not included in this report.

Summarizing Unique Sentences with TF-IDF Modeling

Shifting gears, we applied a more advanced model to look at the differences between these two corpora at a sentence level. Using TF-IDF (a tool that upweights words by how often they occur in a document AND by how often they occur in a corpus) we can search for word importance. By conducting this search on a sentence level (where each sentence is considered its own document within a corpus (all the sentences that candidate said)) we can find the top five most "important" sentences in each of their speeches by assigning word importance based on how often each term occurs in the sentence and how unique it is across all sentences in their debate statements.

This could be a useful tool for if you missed the debate but wanted to know what was said that was important and not just what was *popularly quoted:* not what *gets a lot of views on TikTok* or what *the legacy media channels (all partisan) want you to hear* but the actual **rarest** and **most-unique-to-candidate** sentences they said - by the data.

This matters because we know both candidates want to win over the majority of votes, so they will try to say the most commonly agreeable things to the bulk of the American populace. But, as an undecided voter, I want to know what *different* things they said. Maybe in these differences, I could discern what I really do/don't like (unique to a candidate) that would sway my vote. In a way, using TF-IDF like this results in a *kind* of a summary tool but mainly more of a "summarize the interesting/unique things they said" tool. (Figures 7 and 8).

Experimental - Using Word2Vec to Create ML-Ready Vectors

To finish our language model analysis, we employed Word2Vec using the NLP Python library Gensim. By default, Word2Vec models use skip-gram with negative sampling. Using a window size that we set (we chose three words) W2V creates vectors both for target words and surrounding words and assesses their relatedness, nudging these vectors toward/away from each other until their error is as close to zero as possible, compared to their actual proximity label. Negative sampling is a method where W2V includes non-nearby words (not in the window of three words *before*, or three words *after* the target word) to train the model on what is NOT contextual.

By creating only vectors for main words and their nearby context words, it avoids the sparse matrices we saw in the output of TF-IDF and creates more "dense" information – thus making it better for computational load and larger datasets. Using the corpus without stopwords, this produced a tool where, for Biden or Trump, we could input a word and W2V would give us the top 10 most "related" words in each of their corpora (Figure 9). We chose to eliminate stopwords in hopes that related words would be more topical and less often connecting words/pronouns/simple verbs.

This tool proved to have limited functionality so we went a step further and used clustering methods from sklearn to cluster all of each candidate's vocabularies into 5 clouds of related vectors. We were hitting our technical ceiling on this and decided to wrap up by reducing the vectors involved with Principal Component Analysis so we could create 2D visualizations of these clouds, which, given vocabulary size and the stripped down dimensionality – also proved to be of low-use. (Figures 10 and 11).

Analysis of Word2Vec

W2V definitely becomes more of a black box. It pulls much more nuanced and informative (numeric) data from the debate at the cost of interpretability. We are not able to comprehend the 50-dimension hypersphere that performs this elaborate dance to perfectly update all the vectors within this unfathomable vector-space so they are perfectly aligned amongst each other based on this one corpus. For this reason, it may be

a better and more accurate model with more ML capabilities but for the size of our corpora and the simplicity of our investigation it is "cool" but it doesn't tell us a ton that is really insightful (at least at our level of understanding and coding ability). We concluded that it would probably perform very well with a really large corpus that has *more opportunities to pick up on repeating and/or nuanced patterns* and would be a fantastic tool if we were building bigger, more complex language models with generative or predictive capacity. But for the purpose of this project, we learned more from the bigram model compared to, for example: knowing that when Biden says the word "affordable" that this term is is .41982281 similar to the term "determine." We are both excited to learn more about how to deploy these models in a way that leads us to greater and more actionable insight and it was a great first foray into this caliber of language processing tools.

Conclusion

One of the reasons that data science/analytics is important is that it helps separate objectivity from subjectivity. Dylan and I both agree that, going into all three of these Trump elections, we had prior bias and expectations on what each candidate was going to say and how they were going to act. This is an unfortunate and inherent truth of a two-camp, "us vs. them" political party system with heavily monetized and pervasive influence from mass media. No matter which side of the aisle you sit on, our political system is becoming further and further polarized toward far-left and far-right and this has consequences. It becomes a war instead of a debate; legacy media vs. social media; a competition instead of a conversation. Thus, voters come into these dialogues already heated – ready to hear what they want/expect, both the good and the bad. Running language tools clears the fog of emotions and expectations so we can purely look at frequencies, patterns, averages, presences and absences. Through this exercise, we were able to extract "What did they say and how did they say it?" instead of focusing on "I hate this guy," or "He's too old, he can't remember anything." This clarity is why we do data analytics.

Figures Below

Figure 1A

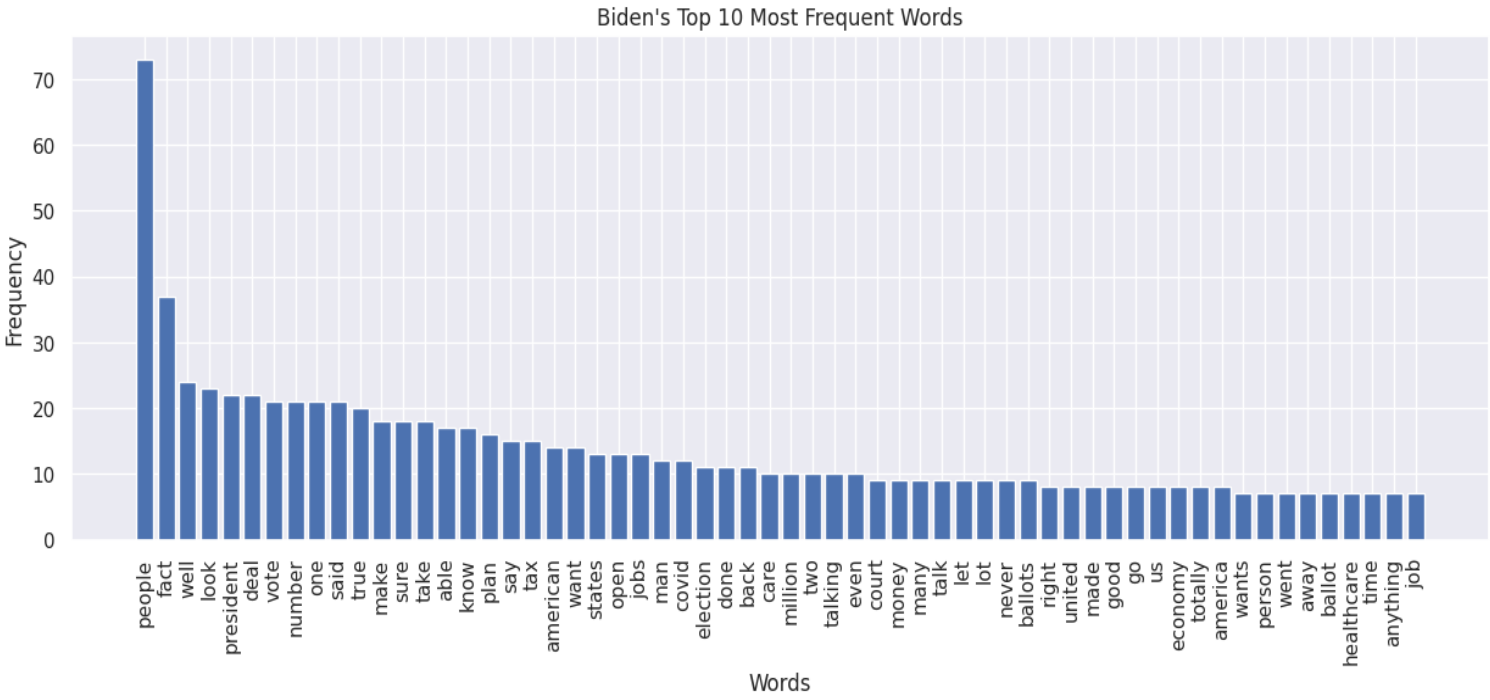
Biden's Top 10 Most Frequent Words
Frequency
0
10
20
30
40
50
60
70
people
fact
well
look
president
deal
vote
number
one
said
true
make
sure
take
able
know
plan
say
tax
american
want
states
open
jobs
man
covid
election
done
back
care
million
two
talking
even
court
money
many
talk
let
lot
never
ballots
right
united
made
good
go
us
economy
totally
america
wants
person
went
away
ballot
healthcare
time
anything
job
Words

Figure 1B

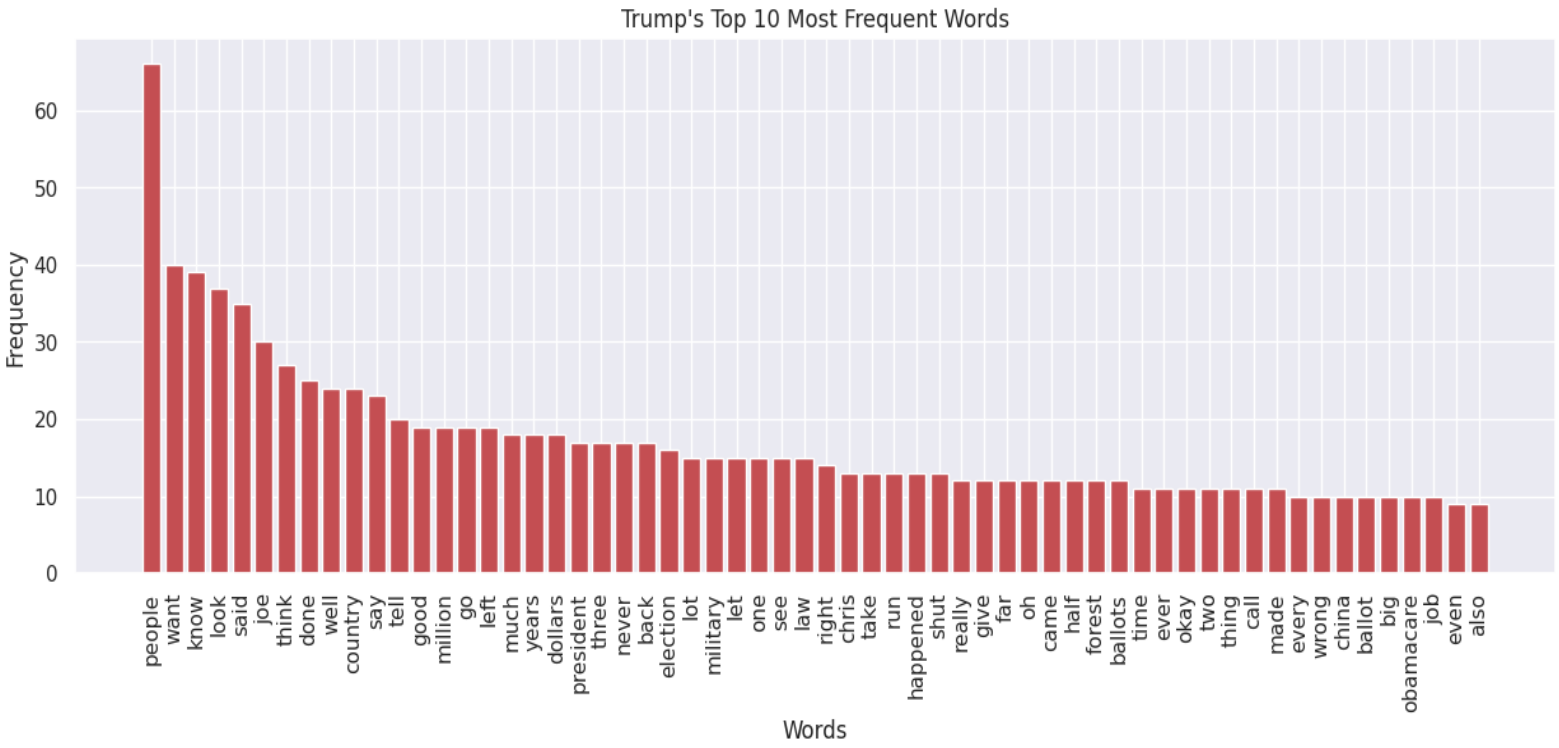

Figure 2

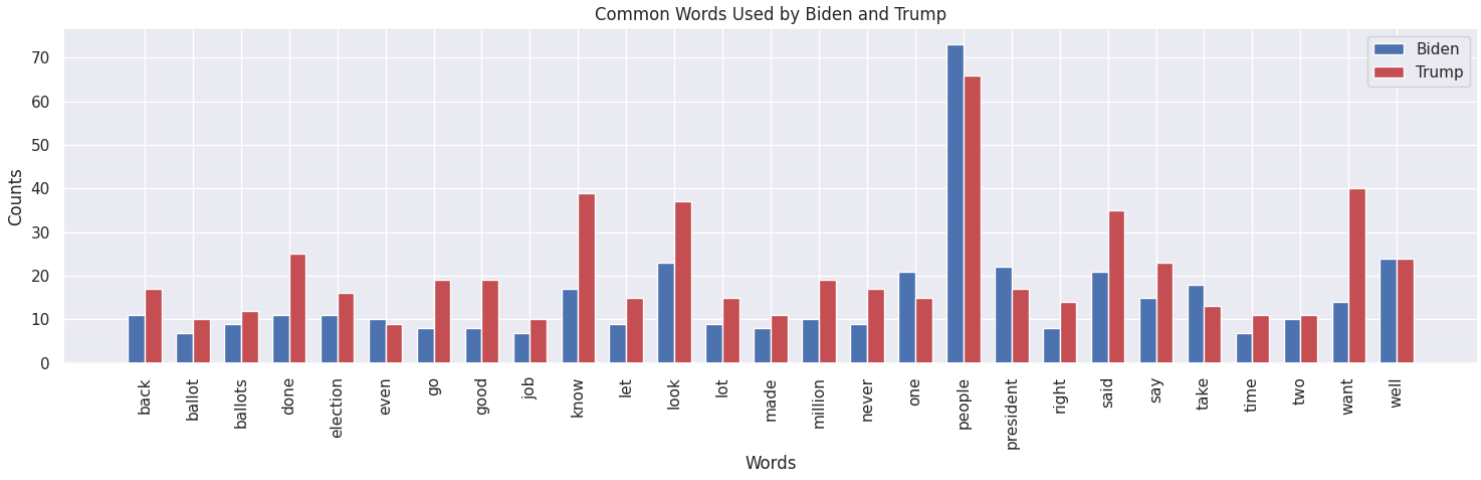

Figure 3

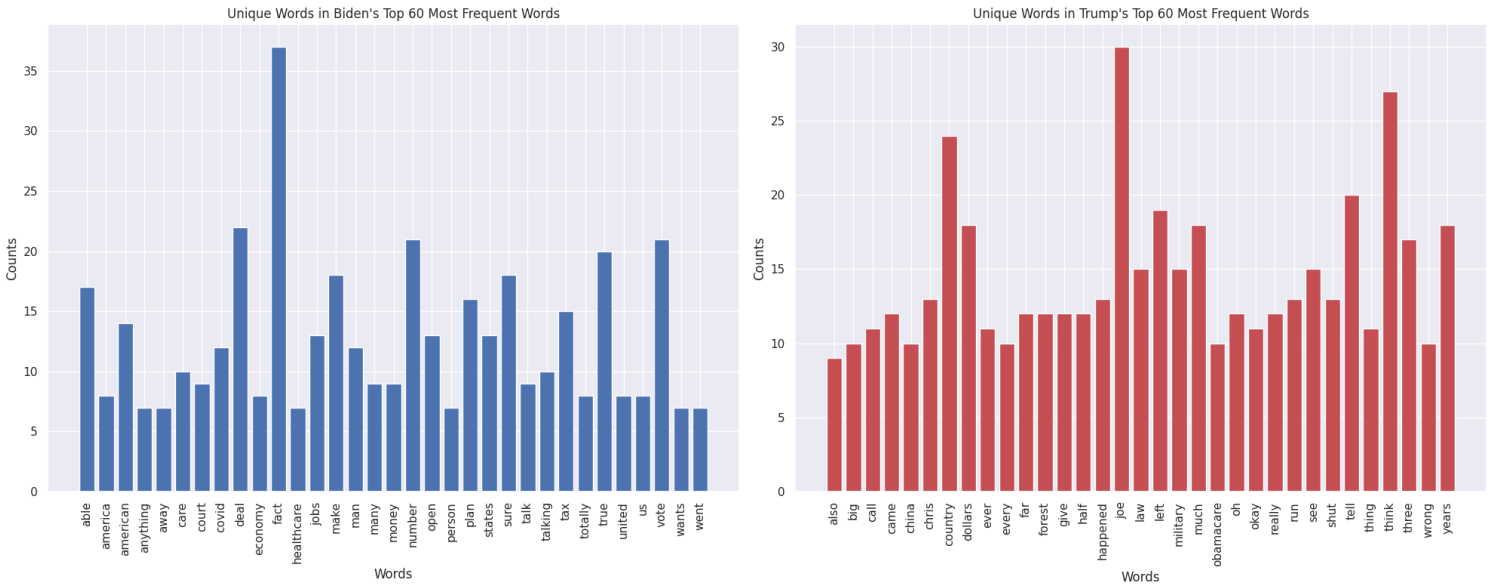

Figure 4

```
[71] # for fun :) we also can generate text, given a starting word
     #this is back to our WITH STOPWORDS processed corpus (how we built our model) so we can get coherent sentences
     biden_generated_text = biden_lm.generate(20, text_seed=['people']) #play around with this, by changing the seed word from their vocab
     print("Biden will probably say:", ' '.join(biden_generated_text))

     Biden will probably say: and let your golf course and so far away from having to eliminate those conditions </s> relaxed fuel economy that

[70] #and here is for trump
     trump_generated_text = trump_lm.generate(20, text_seed=['people'])
     print("Trump will probably say:", ' '.join(trump_generated_text))

     Trump will probably say: will destroy our country </s> whatsoever </s> let me ask you know hunter </s> tell that are far as the
```

Figure 4 shows the bigram model text generation. It is interesting to note that, in many of the repeated trials, and amongst different seed words – Biden's model usually created longer and more coherent sentences. This is preliminary data that *could* show that Biden speaks more fluidly, as simple models are able to learn his speech patterns more recognizably.

Figure 5

```
#back to our WITH STOPWORDS corpus for realistic sentence generation
#trigram model text generation - biden
biden3_generated_text = biden3_lm.generate(10, text_seed=['president, economy']) #play around with this, by changing the seed words
print("Biden will probably say:", ' '.join(biden3_generated_text))
```

```
Biden will probably say: being able to vote because there are million people getting
```

```
#and here is for trump
trump3_generated_text = trump3_lm.generate(10, text_seed=['president, china']) #play around with this, by changing word 1 and 2
print("Trump will probably say:", ' '.join(trump3_generated_text))
```

```
Trump will probably say: that we have the old slugs out there that are
```

Figure 5 shows the trigram model, with smoother text coherence and the hilarity of, even in our language model, Trump is calling people "old slugs."

Figure 6

```
#just for fun, let's see what trump would say in a unigram model
trump1_generated_text = trump1_lm.generate(10, text_seed=['people'])
print("Trump will probably say:", ' '.join(trump1_generated_text))
```

```
Trump will probably say: point got urging football school russia never good the
```

And, just to showcase why unigram models are weak, comparatively, we can see Trump's string of single-probability words that means nothing.

Figure 7

Biden's Unique Sentences: / / It is true, the reason i got in the race is when those people... close your eyes, remember what those people look like coming out of the fields, carrying torches, their veins bulging, just spewing anti-semitic bile and accompanied by the ku klux klan. / / He said, maybe we should drop a nuclear weapon on them, and they may- yeah, he did say that- and here's the deal- ... we're going to be in a position where we can create hard, hard, good jobs by making sure the environment is clean, and we all are in better shape. / / There's many people today driving their kids to soccer practice and/or black and white and hispanic in the same car as there have been any time in the past, what really is a threat to the suburbs and their safety is his failure to deal with covid. / / We're going to build a economy that in fact is going to provide for the ability of us to take 4 million buildings and make sure that they in fact are weatherized in a way that in fact will they'll emit significantly less gas and oil because the heat will not be going out. / / Number two, if in fact, during our administration in the recovery act, i was in charge able to bring down the cost of renewable energy to cheaper than are as cheap as coal and gas and oil.

Figure 7 shows our TF-IDF model's top five most unique sentences from Biden's text. Yikes, even in 2020 you can see Biden's sentence structure and ability to remember his script failing. But we can also see that he really tries to focus on the issues, even if too many per sentence, and in a scatter-brained way. He emphasizes things he wants to do and good things about the present.

Figure 8

Trump's Unique Sentences: The places we had trouble were democratic run cities- i think as a party issue, you can bring in a couple of examples but if you look at chicago, what's going on in chicago where a 53 people were shot and eight died shot, if you look at new york where it's going up, like nobody's ever seen anything. / / The numbers are going up a 100%, 150%, 200% crime, it is crazy what's going on and he doesn't want to say law and order because he can't because he'll lose his radical left supporters and once he does that, it's over with. / / And i'll tell you something, some people say maybe the most important by the end of the first term i'll have approximately 300 federal judges and court of appeals judges, 300 and hopefully three great supreme court judges, justices that is a record the likes of which very few people and one of the reasons i'll have so many judges because president obama and him left me 128 judges to fill. / / By my doing it early, in fact, dr. fauci said, "president trump saved thousands of lives." many of your democrat governors said, "president trump did a phenomenal job." we worked with the governor. / / I love counting the votes chris, he's so wrong when he makes a statement like that- excuse me you know it can't be done.

Figure 8 shows our TF-IDF model's top five most unique sentences from Trump's text. Trump is just about as frantic in his speaking as Biden but we can see much more of a focus on the past (vs what Biden wants to do in the future) and there is much more focus on BAD things from the past and great things about himself.

Figure 9

```
#print similar words
biden_similar_words = biden_w2v_model.wv.most_similar("economy") #put whatever word you are interested in here
print(biden_similar_words) #- prints 10 most similar

#trump's model
trump_w2v_model = Word2Vec(sentences=trump_ngram_processed_corpus_nostop, vector_size=50, window=3, min_count=1, workers=4)

#save the model for later use so we don't have to retrain it every time we open this colab file
#trump_w2v_model.save("trump_w2v_model") #this version saves it to the colab environment, which is temporary
#trump_w2v_model.save("/content/drive/MyDrive/word2vec_trump.model") #this one saves it to my google drive

#load the model
#biden_w2v_model = Word2Vec.load("biden_w2v_model") #once again, for the colab space
biden_w2v_model = Word2Vec.load("/content/drive/MyDrive/word2vec_biden.model") #from the permanent drive storage

#print similar words
trump_similar_words = trump_w2v_model.wv.most_similar("economy") #put whatever word you are interested in here
print(trump_similar_words) #- prints 10 most similar
```

```
[('hispanic', 0.2988523244857788), ('three', 0.2863776981830597), ('true', 0.272894948720932), ('insisting', 0.2638814151287079),
[('businesses', 0.419029176235199), ('made', 0.4053993225097656), ('press', 0.36307021975517273), ('crooked', 0.36182254552841187)
```

Figure 9 shows our accurate but hard to interpret word (vector) similarity tool from Word2Vec modeling.

(next page)

Figure 10

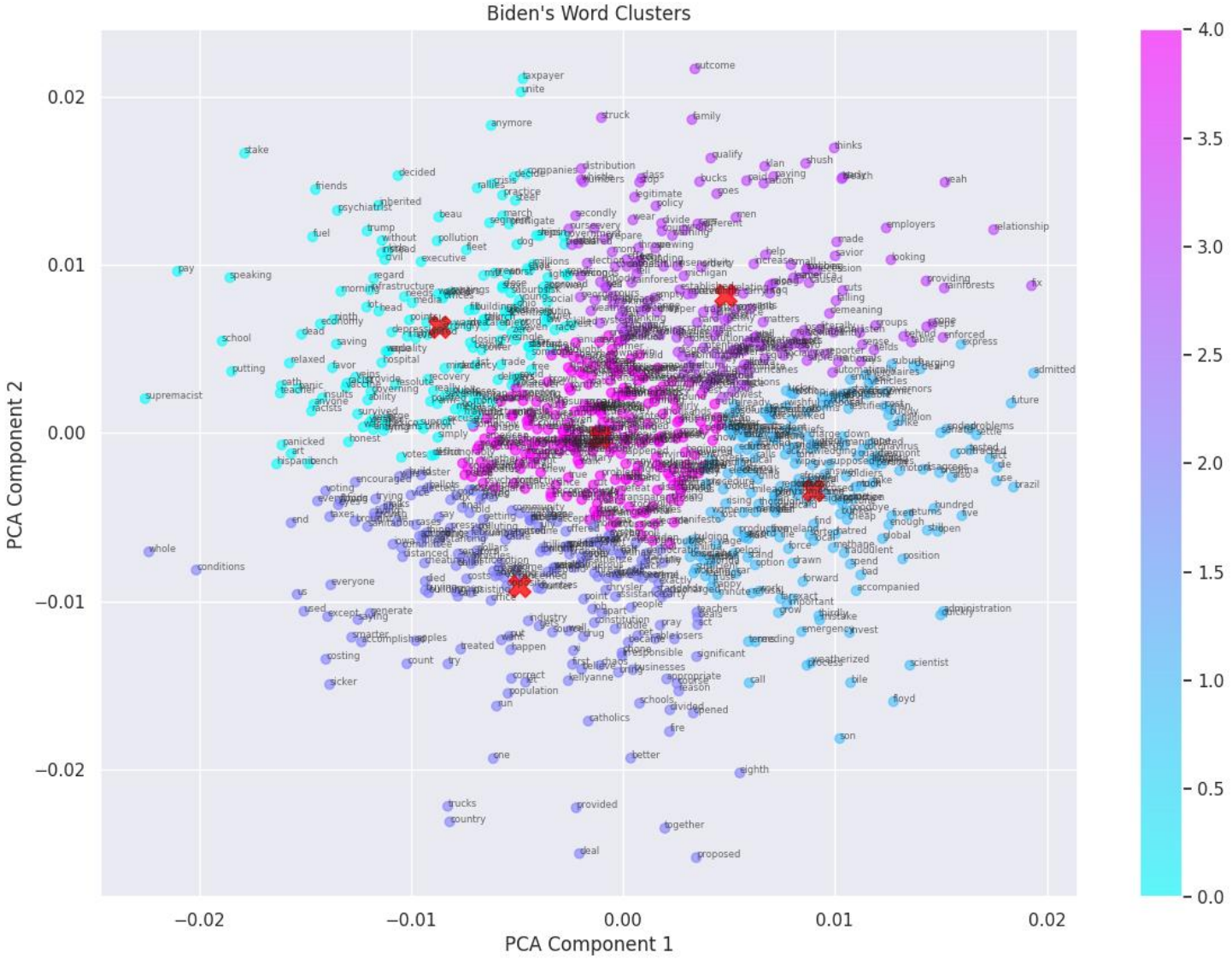

Figure 10 shows Biden's five word clusters. Good luck reading/interpreting this. It was a great exercise but did not yield usable insights.

next page

Figure 11

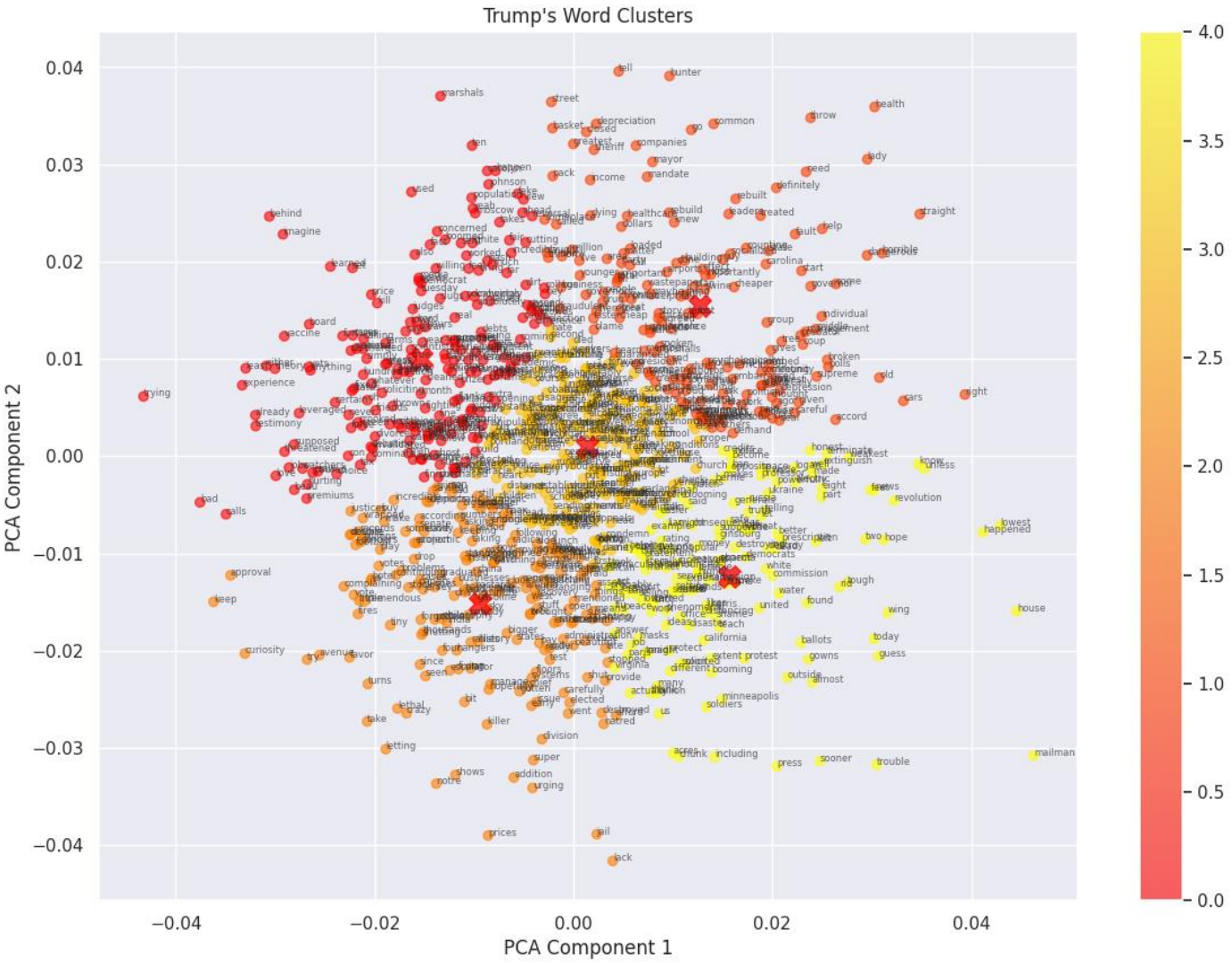

Figure 11 shows Trump's 5 word clusters.