Comparative Text Analysis of U.S. Presidential Inauguration Speeches

I started my analysis by reading in multiple texts. I then settled on the following four texts: Bill Clinton's inauguration speech from January 20th 1993, G. W. Bush's inauguration speech from January 20th 2001, Barack Obama's inauguration speech from January 20th 2009 and lastly Donald Trump's inauguration speech from January 20th 2017. The reason I chose these speeches is because I wanted to analyze two speeches from former Republican presidents and two speeches from former Democratic presidents at a certain important point in their political careers, and what could possibly be more important than their very first inauguration speech? We didn't have Clinton's and Bush's speeches available on Courseworks so I found them on the UVA Miller Center's website. They have an amazing collection of speeches of many former presidents so I highly recommend to check them out if interested. I will compare the texts further throughout this lab.

```
In [2]: import pandas as pd
        import nltk
        import warnings
        import plotly.express as px
        import numpy as np
        import matplotlib.pyplot as plt
        warnings.filterwarnings('ignore')

In [5]: df_Obama_speech = pd.read_csv('Text 1-Obama speech.csv',encoding='cp1252')
        df_Trump_speech = pd.read_csv('Text 1-Trump speech.csv',encoding='cp1252')

n [126]: df_Obama_2013 = pd.read_csv('2013-Obama.txt', on_bad_lines='skip',encoding='utf8')

n [127]: df_Obama_2014 = pd.read_csv('2014-Obama.txt', on_bad_lines='skip',encoding='utf8')

In [12]: df_Obama_2015 = pd.read_csv('2015-Obama.txt', on_bad_lines='skip',encoding='utf8')

In [14]: df_Obama_2016 = pd.read_csv('2016-Obama.txt', on_bad_lines='skip',encoding='cp1252')

n [386]: df_Obama_Inaug = pd.read_csv('Text 3-Obama Inaug.2009.txt', on_bad_lines='skip',encoding='utf8')

n [387]: df_Trump_Inaug = pd.read_csv('Text 3-Trump Inaug.2017.txt', on_bad_lines='skip',encoding='utf8')

n [388]: df_Obama_Sous = pd.read_csv('Text 4-sous-2012-Obama.txt', on_bad_lines='skip',encoding='utf8')

n [389]: df_GWBush_Inaug = pd.read_csv('GWBush-Inaug-2001.txt', on_bad_lines='skip',encoding='utf8')

n [390]: df_Clinton_Inaug = pd.read_csv('Clinton_Inaug_1993.txt', on_bad_lines='skip',encoding='utf8')

n [391]: df_Clinton_1997 = pd.read_csv('1997-Clinton.txt', on_bad_lines='skip',encoding='utf8')

n [375]: df_GWBush_2001 = pd.read_csv('2001-GWBush-1.txt', on_bad_lines='skip',encoding='utf8')

In [20]: df_presidents_sheet = pd.read_csv('Text 5-adverbs-sous-presidents-Sheet1.csv',encoding='cp1252')

In [23]: df_sou_texts = pd.read_csv('Text 5-adverbs-sous-texts.csv',on_bad_lines='skip',encoding='utf8')

In [24]: df_SUBTLEX_PoS = pd.read_csv('Text 5-SUBTLEX_PoS.csv',encoding='cp1252')
```

```
In [398]: df_Clinton_Inaug = pd.read_csv('Clinton_Inaug_1993.txt', on_bad_lines='skip',encoding='utf8')
          df_Clinton_Inaug
```

Out[398]:

	My fellow citizens	today we celebrate the mystery of American renewal. This ceremony is held in the depth of winter	but by the words we speak and the faces we show the world	we force the spring	a spring reborn in the world's oldest democracy that brings forth the vision and courage to reinvent America. When our Founders boldly declared America's independence to the world and our purposes to the Almighty	they knew that America	to endure	would have to change; not change for change's sake but change to preserve America's ideals: life	liberty	the pursuit of happiness. Though we marched to the music of our time	our mission is timeless. Each generation of Americans must define what it means to be an American.
0	On behalf of our Nation	I salute my predecessor	President Bush	for his half-century of service to America. A...	fascism	and communism.	NaN	NaN	NaN	NaN	NaN
1	Today	a generation raised in the shadows of the col...	we inherit an economy that is still the world...	stagnant wages	increasing inequality	and deep divisions among our own people.	NaN	NaN	NaN	NaN	NaN
2	When George Washington first took the oath I h...	news traveled slowly across the land by horse...	the sights and sounds of this ceremony are br...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	We earn our livelihood in America today in pea...	great and small; when the fear of crime robs ...	we have not made change our friend.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	Our democracy must be not only the envy of the...	and a new season of American renewal has begun.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	To renew America	we must be bold. We must do what no generatio...	in their jobs	and in their future	and at the same time cut our massive debt. An...	but it can be done and done fairly	not choosing sacrifice for its own sake but f...	NaN	NaN	NaN	NaN
	Our Founders	from whom	and to whom we bear								

```
In [392]: df_GWBush_Inaug = pd.read_csv('GWBush-Inaug-2001.txt', on_bad_lines='skip',encoding='utf8')
```

click to unscroll output; double click to hide

Out[392]:

	President Clinton	distinguished guests and my fellow citizens	the peaceful transfer of authority is rare in history	yet common in our country. With a simple oath	we affirm old traditions and make new beginnings.
0	As I begin	I thank President Clinton for his service to ...	NaN	NaN	NaN
1	And I thank Vice President Gore for a contest ...	NaN	NaN	NaN	NaN
2	I am honored and humbled to stand here	where so many of America's leaders have come ...	and so many will follow.	NaN	NaN
3	It is the American story—a story of flawed and...	united across the generations by grand and en...	NaN	NaN	NaN
4	The grandest of these ideals is an unfolding A...	that everyone deserves a chance	that no insignificant person was ever born.	NaN	NaN
5	Americans are called to enact this promise in ...	and sometimes delayed	we must follow no other course.	NaN	NaN
6	Through much of the last	America's faith in freedom and	taking root in many nations.	NaN	NaN

```
In [395]: df_Obama_Inaug = pd.read_csv('Text 3-Obama Inaug.2009.txt', on_bad_lines='skip',encoding='utf8')
```

click to scroll output; double click to hide

Out[395]:

	Vice President Biden	Mr. Chief Justice	Unnamed: 2
0	members of the United States Congress	distinguished guests	and fellow citizens:
1	Each time we gather to inaugurate a President ...	NaN	NaN
2	And for more than two hundred years	we have.	NaN
3	Through blood drawn by lash and blood drawn by...	we learned that no union founded on the princ...	and vowed to move forward together.
4	Together	we determined that a modern economy requires ...	schools and colleges to train our workers.
5	Together	we discovered that a free market only thrives...	NaN
6	Together	we resolved that a great nation must care for...	and protect its people from life's worst haza...
7	We will defend our people and uphold our value...	but because engagement can more durably lift ...	NaN
8	Thank you. God bless you	and may He forever bless these United States ...	NaN

```
In [394]: df_Trump_Inaug = pd.read_csv('Text 3-Trump Inaug.2017.txt', on_bad_lines='skip',encoding='utf8')
```

click to scroll output; double click to hide

Out[394]:

	Chief Justice Roberts	President Carter	President Clinton	President Bush	President Obama	fellow Americans	and people of the world: thank you.
0	We	the citizens of America	are now joined in a great national effort to ...	NaN	NaN	NaN	NaN
1	Together	we will determine the course of America and t...	NaN	NaN	NaN	NaN	NaN
2	We will face challenges. We will confront hard...	NaN	NaN	NaN	NaN	NaN	NaN
3	Every four years	we gather on these steps to carry out the ord...	and we are grateful to President Obama and Fi...	NaN	NaN	NaN	NaN
4	Today's ceremony	however	has very special meaning. Because today we ar...	or from one party to another – but we are tra...	D.C. and giving it back to you	the American People.	NaN
...	...	...	...	...	...	...	...
69	Together	We Will Make America Strong Again.	NaN	NaN	NaN	NaN	NaN
70	We Will Make America Wealthy Again.	NaN	NaN	NaN	NaN	NaN	NaN
71	We Will Make America Proud Again.	NaN	NaN	NaN	NaN	NaN	NaN
72	We Will Make America Safe Again.	NaN	NaN	NaN	NaN	NaN	NaN
73	And	Yes	Together	We Will Make America Great Again. Thank you	God Bless You	And God Bless America.	NaN

74 rows × 7 columns

Creating bag-of-words representations by cleaning and tokenizing the speech texts

I will now create bag of words out of my text data. Bag of words are a simple way to transform text data into numerical data in order to use it for our analysis moving forward. As seen below, I combined the data preprocessing and bag of words conversion into one code line:

The data preprocessing is done through lower casing, removing punctuation, removing stop words, removing extra white space and stemming. We do that in order to remove the "extra noise" around our data. No valuable information goes missing through the process which is great.

The bag of word model was created with the NLTK library (there are multiple ways to create a model, for example with CountVector). The set is stored in the variable "vocab" as it contains all the individual words in our list that's called "text_data". This is important for our next step as every single word in every single text document in our "text_data" list is being counted and then stored in and represented by dictionaries. These dictionaries then are being stored as a list in the "bow_model" variable.

In [18]:
```
import nltk
import string
import re

def WordPreProcessing(text_data):

    # create the vocabulary
    vocab = set()

    # create the bag-of-words model
    bow_model = []

    for text in text_data:
        #print("text : " + text)
        # create a dictionary to store the word counts
        word_counts = {}

        # remove leading and trailing white space
        text = text.strip()

        # replace multiple consecutive white space characters with a single space
        text = " ".join(text.split())

        #print("remove spaces : "+text)
        # tokenize the text
        tokens = nltk.word_tokenize(text)
        #print (tokens)
        # lowercase the tokens
        lowercased_tokens = [token.lower() for token in tokens]

        #print("lower Case : " + str (lowercased_tokens))
        # remove punctuation
        filtered_tokens = [token for token in lowercased_tokens if token not in string.punctuation]

        #print("filtered_punct :" + str (filtered_tokens))
        # get list of stopwords in English
        stopwords = nltk.corpus.stopwords.words("english")

        # remove stopwords
        filtered_tokens = [token for token in filtered_tokens if token not in stopwords]

        # remove non english charcter in words
        filtered_tokens = [re.sub(r'[^a-zA-Z0-9\s]+','',token) for token in filtered_tokens]


        #print("filtered_stopwords :" + str (filtered_tokens))
        # create stemmer object
        stemmer = nltk.stem.PorterStemmer()

        # stem each token
        stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]

        # remove empty words
        last_stage = [token for token in stemmed_tokens if token]

        #print("stemmed_tokens :" + str (stemmed_tokens))
        # update the vocabulary
        vocab.update(last_stage)

        # count the occurrences of each word
        for word in last_stage:
            if word in word_counts:
                word_counts[word] += 1
            else:
                word_counts[word] = 1

        # add the word counts to the bag-of-words model
        bow_model.append(word_counts)
    return bow_model
```

In [19]:
```
text_data_clinton = df_Clinton_Inaug
Clinton_Speech_BOW = WordPreProcessing(text_data_clinton)
```

In [20]:
```
text_data_bush = df_GWBush_Inaug
Bush_Speech_BOW = WordPreProcessing(text_data_bush)
```

In [21]:
```
text_data_obama = df_Obama_Inaug
Obama_Speech_BOW = WordPreProcessing(text_data_obama)
```

In [22]:
```
text_data_trump = df_Trump_Inaug
Trump_Speech_BOW = WordPreProcessing(text_data_trump)
```

Below you can see the actual bag of words for Trump's speech. You can see it as a list with dictionaries stored inside (separated by the curly brackets). Every element has a key (word) and a value count (number):

```
In [90]: Trump_Speech_BOW

Out[90]: [{'citizen': 1,
           'america': 1,
           'join': 1,
           'great': 1,
           'nation': 1,
           'effort': 1,
           'rebuild': 1,
           'countri': 1,
           'restor': 1,
           'promis': 1,
           'peopl': 1},
          {'determin': 1, 'cours': 1, 'america': 1, 'world': 1, 'year': 1, 'come': 1},
          {},
          {'gather': 1,
           'step': 1,
           'carri': 1,
           'orderli': 1,
           'peac': 1,
           'transfer': 1,
```

Calculating and comparing relative word frequencies across speeches

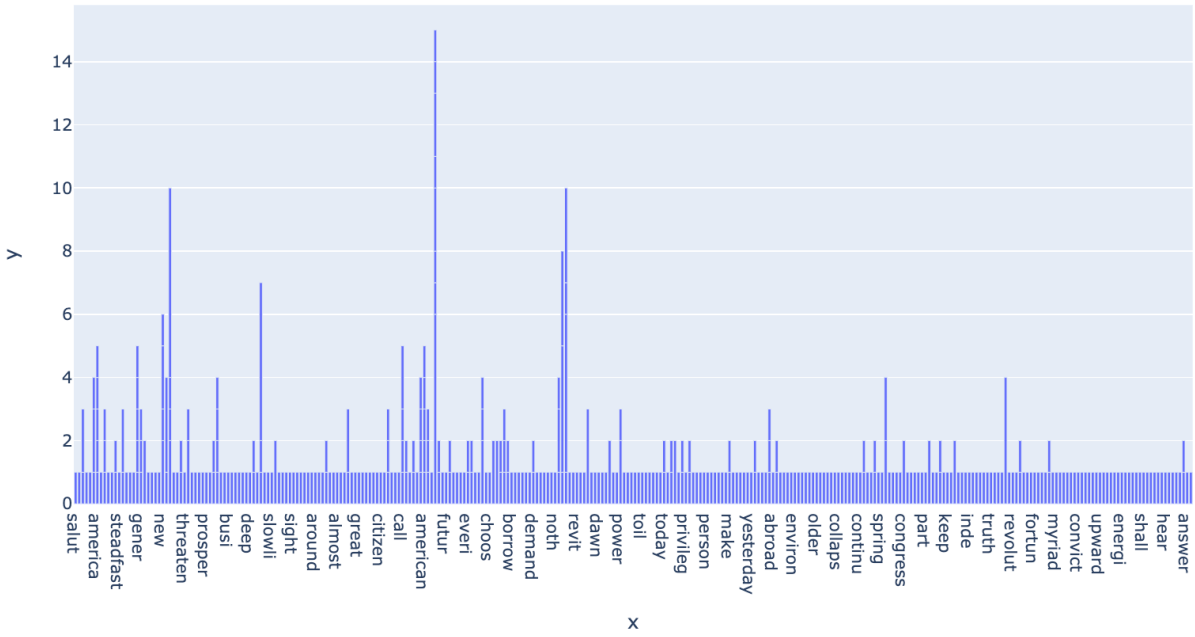
Next up I will generate the relative word frequencies for the bag of words as this will make it very convenient to compare the bag of words under one another. This is generally a great method to look if there are any key themes or patterns in one of our speeches or just to gain some insight about the general content. Below you can see that I visualized the overall word frequencies for each bag of words individually with a bar chart and then the top 15 most frequent words of every bag of words together in one table. Let's start with the bar charts:

Looking at the diversity of the relative word frequency of each bag of words (i.e., how many different words were used more than once during a speech), it is interesting to see that Barack Obama's speech is by far the least diverse one whereas Bill Clinton's seems to be the most diverse one. G.W. Bush and Donald Trump are both in the middle and relatively similar to one another when it comes to their speeches.

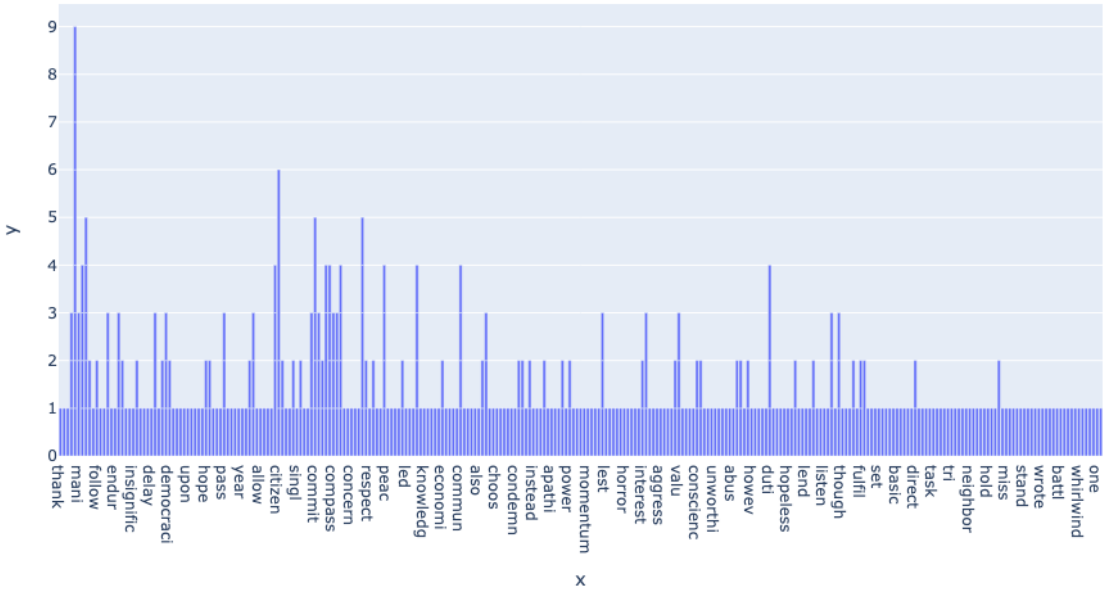
```
def display_Frequencies(speech):
    words = []
    count = []
    for i in range(len(speech)):
        for keys in speech[i].keys():
            words.append(keys)
        for values in speech[i].values():
            count.append(values)
    fig = px.bar(x=words, y=count)
    fig.show()
```

Clinton 1993

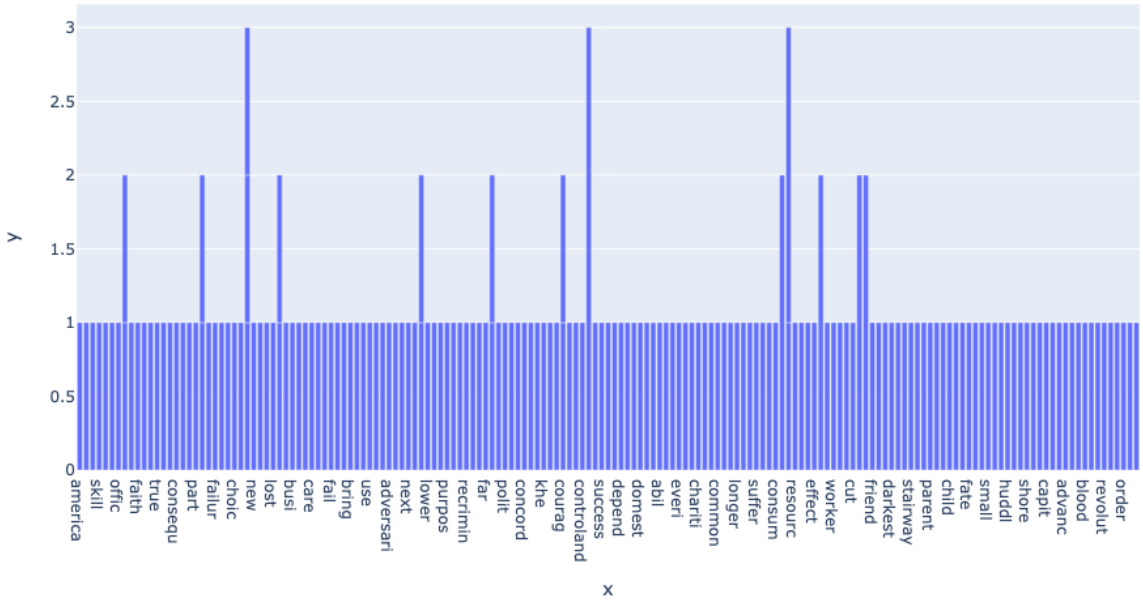
```
display_Frequencies(Clinton_Speech_BOW)
```



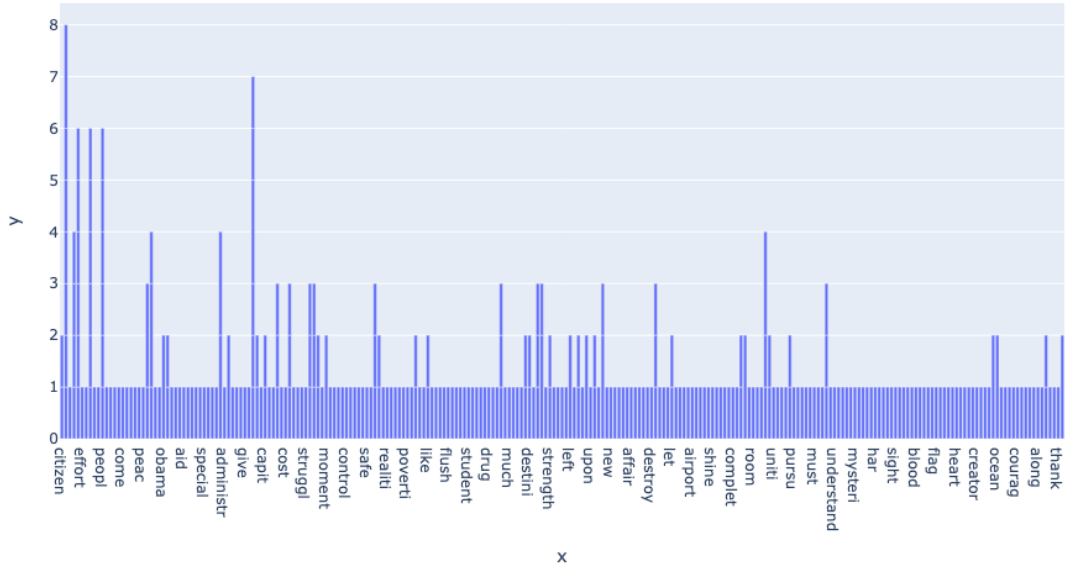
G.W. Bush 2001



Obama 2006



Trump 2017



Looking at the word frequencies in more detail, it shows further that Bill Clinton's speech also has the highest word counts per word (with multiple words counted 4 times upwards to 6 times) compared to the other speeches. Trump, in comparison, never uses a word more than twice in his speech.

I think in order to draw any conclusions in regard to the context of a speech (i.e., key topic) it is worth to look at each frequency table individually: For Bill Clinton's speech for example, we already know through the bar charts that he uses many words in his speech at least twice. I would therefore only pay attention to the words that he uses at least four times during his speech. These words are world, must, us, peopl, let, idea.

G.W. Bush's speech is a little harder to contextualize because even though he uses many words more than twice in his speech, that is essentially it. The only word he actually uses very prominently is the word "citizen".

Even though the least diverse one when it comes to word frequency of the whole speech, Barack Obama's speech is now again a bit easier to contextualize because there are such few words that he uses more than once: prosper, world, see, hour, chang.

Lastly, Donald Trump's speech is probably the hardest to conceptualize out of the four because, as mentioned before, he does have some diversity in word frequency in his speech but he doesn't use any word more than twice. What helps a little bit in his speech though is that he uses words that are very easy to conceptualize in and of itself, for example "America", "god", "bless", "power" and "loyalty".

```
def Return_df(speech):
    words = []
    count = []
    for i in range(len(speech)):
        for keys in speech[i].keys():
            words.append(keys)
        for values in speech[i].values():
            count.append(values)
    return pd.DataFrame({"words":words,"variable Count":count})
```

```
# Transform bag of words into data frame
Clinton_speech_df = Return_df(Clinton_Speech_BOW)
Bush_speech_df = Return_df(Bush_Speech_BOW)
Obama_speech_df = Return_df(Obama_Speech_BOW)
Trump_speech_df = Return_df(Trump_Speech_BOW)
```

```
TopTenC = Clinton_speech_df.sort_values(['variable Count'], ascending=False).iloc[:15,]
TopTenC = TopTenC.reset_index(drop=True)
TopTenC = TopTenC.rename(columns={"words": "Clinton", "variable Count": "Count"})
TopTenB = Bush_speech_df.sort_values(['variable Count'], ascending=False).iloc[:15,]
TopTenB = TopTenB.reset_index(drop=True)
TopTenB = TopTenB.rename(columns={"words": "Bush", "variable Count": "Count"})
TopTenO = Obama_speech_df.sort_values(['variable Count'], ascending=False).iloc[:15,]
TopTenO = TopTenO.reset_index(drop=True)
TopTenO = TopTenO.rename(columns={"words": "Obama", "variable Count": "Count"})
TopTenT = Trump_speech_df.sort_values(['variable Count'], ascending=False).iloc[:15,]
TopTenT = TopTenT.reset_index(drop=True)
TopTenT = TopTenT.rename(columns={"words": "Trump", "variable Count": "Count"})
```

```
result = pd.concat([TopTenC, TopTenB, TopTenO, TopTenT], axis=1)
result
```

	Clinton	Count	Bush	Count	Obama	Count	Trump	Count
0	world	6	citizen	4	prosper	3	bless	2
1	must	6	us	2	world	2	american	2
2	us	5	spirit	2	s	2	ocean	2
3	peopl	5	new	2	see	2	industri	2
4	let	4	work	2	hour	2	obama	2
5	idea	4	civil	2	chang	2	moment	2
6	capit	3	common	2	selfless	1	left	2
7	us	3	purpos	2	s	1	one	2
8	let	3	find	2	resourc	1	transfer	2
9	respons	3	must	2	without	1	power	2
10	still	2	duti	2	regard	1	one	2
11	work	2	nation	2	effect	1	million	2
12	power	2	mani	2	must	1	loyalti	2
13	new	2	commit	2	america	1	america	2
14	place	2	human	1	worker	1	god	2

Observing and interpreting differences in word usage, connecting patterns to political context and party affiliation

I expected to see that the bag of words of Bill Clinton and Barack Obama would be similar as well as the ones of G.W. Bush and Donald Trump. My expectations were based off of the theory that being elected by the same party voters indicates that you care about the same topics as a president. Consequently, I thought that the speeches between the different parties will differ a lot.

What actually came out during the analysis of the word of bags frequency is that in general all four speeches differ a lot from each other, no matter if it comes to the diversity of relative word frequencies or frequently used words in detail. There is really not much similarity at all which is a little shocking to me.

Comparing speeches using statistical metrics like cosine similarity to measure linguistic overlap and divergence

I will now compare the texts through the very popular cosine similarity. Cosine similarity is used to compare texts semantically in size.

I had problems with running my code using my bag of words from the analysis so I decided to recreate the bag of words again in a different way that included the function for cosine similarity in one code line which ended up working. I did it as follows:

```
W = pd.read_csv('Clinton_Inaug_1993.txt', on_bad_lines='skip',encoding='utf8')
W = W[W.columns[1:]].apply(
    lambda x: ','.join(x.dropna().astype(str)),
    axis=1
)
W = W.to_string().lower()
W
```

```
'0      i salute my predecessor, president bush, for ...\n1      a generation raised in the shadows of the col...\n
2      news traveled slowly across the land by horse...\n3      great and small; when the fear of crime robs ...\n4
and a new season of american renewal has begun.\n5      we must be bold. we must do what no generatio...\n6      fr
om whom we have borrowed our planet, and to...\n7      we must revitalize our democracy. this beauti...\n8      we
must meet challenges abroad as well as at ...\n9      the presidency, and the political process its...\n10     you,
too, must play your part in our renewal....\n11     an idea born in revolution and renewed throug...\n12     my fel
low americans, as we stand at the edge ...\n13                                  and god bless you all.'
```

```
X = pd.read_csv('GWBush-Inaug-2001.txt', on_bad_lines='skip',encoding='utf8')
X = X[X.columns[1:]].apply(
    lambda x: ','.join(x.dropna().astype(str)),
    axis=1
)
X = X.to_string().lower()
X
```

```
"0      i thank president clinton for his service to ...\n1                                                    \n
2      where so many of america's leaders have come ...\n3      united across the generations by grand and en...\n4
that everyone deserves a chance, that no insi...\n5      and sometimes delayed, we must follow no othe...\n6      a
merica's faith in freedom and democracy was ...\n7      it is the inborn hope of our humanity, an ide...\n8      an
d we will not allow it. our unity, our unio...\n9                                                      \n10
\n11     we affirm a new commitment to live out our na...\n12     at its best, matches a commitment to princip
l...\n13     in a time of peace, the stakes of our debates...\n14     it will not be led. if we do not turn the he
a...\n15     of community over chaos. and this commitment,...\n16                    at its best, is also courageo
us.\n17     when defending common dangers defined our com...\n18     we will reclaim america's schools, before ig
n...\n19     sparing our children from struggles we have t...\n20                    lest weakness invite challen
ge.\n21        so that a new century is spared new horrors.\n22     shaping a balance of power that favors freed
o...\n23     at its best, is compassionate. in the quiet o...\n24     we can agree that children at risk are not a
t...\n25     however necessary, is no substitute for hope ...\n26     there is duty. americans in need are not str
a...\n27     for civil rights and common schools, yet comp...\n28     synagogue and mosque lend our communities th
```

```
Y = pd.read_csv('Obama2009.txt', on_bad_lines='skip',encoding='cp1252')
Y = Y[Y.columns[1:]].apply(
    lambda x: ','.join(x.dropna().astype(str)),
    axis=1
)
Y = Y.to_string().lower()
Y
```

```
'forty-four americans have now taken the presidential oath. the words have been spoken during rising tides of prosp
erity and the still waters of peace. yet
america has carried on not simply because of ...\nso it has been. so it must be with this generation of americans.
\nthat we are in the midst of crisis is now well understood. our nation is at war
a consequence of greed and irresponsibility o...\nthese are the indicators of crisis
and that the next generation must lower its s...\ntoday i say to you that the challenges we face are real. they are
```

```
Z = pd.read_csv('Text 3-Trump Inaug.2017.txt', on_bad_lines='skip',encoding='utf8')
Z = Z[Z.columns[1:]].apply(
    lambda x: ','.join(x.dropna().astype(str)),
    axis=1
)
Z = Z.to_string().lower()
Z
```

```
'0      the citizens of america, are now joined in a ...\n1      we will determine the course of america and t...\n
2                                                  \n3      we gather on these steps to carry out the ord...\n4
however, has very special meaning. because to...\n5      a small group in our nation’s capital has rea...\n6
\n7                              and the factories closed.\n8                  but not the citizens of our countr
y.\n9      there was little to celebrate for struggling ...\n10     and right now, because this moment is your m
o...\n11                                                  \n12
\n13       the united states of america, is your country.\n14     but whether our government is controlled by
t...\n15     will be remembered as the day the people beca...\n16
\n17                                                  \n18
\n19                                                  \n20     safe neighborhoods for their families, and g
o...\n21                                                  \n22     a different reality exists: mothers and chil
d...\n23                                                  \n24                  one home, and one glorious desti
ny.\n25                                                 \n26     we’ve enriched foreign industry at the expen
s...\n27                                                  \n28
```

I compared the speeches all one by one below. A cosine similarity can be between 0 and 1 meaning 0 that there is no similarity at all and 1 means entirely similar. We can see that the speeches are

not very similar to another all throughout which confirms our findings from questions three and four.

The speeches that were most similar to each other were the speeches of G.W. Bush and Donald Trump (0.287) and Bill Clinton and G.W. Bush (0.235). The speeches that were the least similar were the ones of Clinton and Barack Obama (0.128) and Barack Obama and Donald Trump (0.13).

And here again, I find it very interesting that, despite all of the speeches being generally pretty dissimilar from one another, the speeches of Bill Clinton and Barack Obama are the most dissimilar of them all since they are both Democrats. I would have expected the speeches of politicians from the same party to be very similar.

```
# Clinton vs Bush
# tokenization
W_list = word_tokenize(W)
X_list = word_tokenize(X)

# sw contains the list of stopwords
sw = stopwords.words('english')
l1 =[];l2 =[]

# remove stop words from the string
W_set = {w for w in W_list if not w in sw}
X_set = {w for w in X_list if not w in sw}

# form a set containing keywords of both strings
rvector = W_set.union(X_set)
for w in rvector:
    if w in W_set: l1.append(1) # create a vector
    else: l1.append(0)
    if w in X_set: l2.append(1)
    else: l2.append(0)
c = 0

# cosine formula
for i in range(len(rvector)):
        c+= l1[i]*l2[i]
cosine = c / float((sum(l1)*sum(l2))**0.5)
print("similarity: ", cosine)
```

```
similarity:  0.23495144355142164
```

```
# Bush vs Obama
# tokenization
X_list = word_tokenize(X)
Y_list = word_tokenize(Y)

# sw contains the list of stopwords
sw = stopwords.words('english')
l1 =[];l2 =[]

# remove stop words from the string
X_set = {w for w in X_list if not w in sw}
Y_set = {w for w in Y_list if not w in sw}

# form a set containing keywords of both strings
rvector = X_set.union(Y_set)
for w in rvector:
    if w in X_set: l1.append(1) # create a vector
    else: l1.append(0)
    if w in Y_set: l2.append(1)
    else: l2.append(0)
c = 0

# cosine formula
for i in range(len(rvector)):
        c+= l1[i]*l2[i]
cosine = c / float((sum(l1)*sum(l2))**0.5)
print("similarity: ", cosine)
```

```
similarity:  0.15839698777049716
```

```
# Obama vs Trump
# tokenization
Y_list = word_tokenize(Y)
Z_list = word_tokenize(Z)

# sw contains the list of stopwords
sw = stopwords.words('english')
l1 =[];l2 =[]

# remove stop words from the string
Y_set = {w for w in Y_list if not w in sw}
Z_set = {w for w in Z_list if not w in sw}

# form a set containing keywords of both strings
rvector = Y_set.union(Z_set)
for w in rvector:
    if w in Y_set: l1.append(1) # create a vector
    else: l1.append(0)
    if w in Z_set: l2.append(1)
    else: l2.append(0)
c = 0

# cosine formula
for i in range(len(rvector)):
        c+= l1[i]*l2[i]
cosine = c / float((sum(l1)*sum(l2))**0.5)
print("similarity: ", cosine)
```

```
similarity:  0.13003912165257484
```

```
# Obama vs Clinton
# tokenization
Y_list = word_tokenize(Y)
W_list = word_tokenize(W)

# sw contains the list of stopwords
sw = stopwords.words('english')
l1 =[];l2 =[]

# remove stop words from the string
Y_set = {w for w in Y_list if not w in sw}
W_set = {w for w in W_list if not w in sw}

# form a set containing keywords of both strings
rvector = Y_set.union(W_set)
for w in rvector:
    if w in Y_set: l1.append(1) # create a vector
    else: l1.append(0)
    if w in W_set: l2.append(1)
    else: l2.append(0)
c = 0

# cosine formula
for i in range(len(rvector)):
        c+= l1[i]*l2[i]
cosine = c / float((sum(l1)*sum(l2))**0.5)
print("similarity: ", cosine)
```

```
similarity:  0.12818706987301454
```

```
# Clinton vs Trump
# tokenization
W_list = word_tokenize(W)
Z_list = word_tokenize(Z)

# sw contains the list of stopwords
sw = stopwords.words('english')
l1 =[];l2 =[]

# remove stop words from the string
W_set = {w for w in W_list if not w in sw}
Z_set = {w for w in Z_list if not w in sw}

# form a set containing keywords of both strings
rvector = W_set.union(Z_set)
for w in rvector:
    if w in W_set: l1.append(1) # create a vector
    else: l1.append(0)
    if w in Z_set: l2.append(1)
    else: l2.append(0)
c = 0

# cosine formula
for i in range(len(rvector)):
        c+= l1[i]*l2[i]
cosine = c / float((sum(l1)*sum(l2))**0.5)
print("similarity: ", cosine)
```

```
similarity:  0.1840294470674847
```

```
# Bush vs Trump
# tokenization
X_list = word_tokenize(X)
Z_list = word_tokenize(Z)

# sw contains the list of stopwords
sw = stopwords.words('english')
l1 =[];l2 =[]

# remove stop words from the string
X_set = {w for w in X_list if not w in sw}
Z_set = {w for w in Z_list if not w in sw}

# form a set containing keywords of both strings
rvector = X_set.union(Z_set)
for w in rvector:
    if w in X_set: l1.append(1) # create a vector
    else: l1.append(0)
    if w in Z_set: l2.append(1)
    else: l2.append(0)
c = 0

# cosine formula
for i in range(len(rvector)):
        c+= l1[i]*l2[i]
cosine = c / float((sum(l1)*sum(l2))**0.5)
print("similarity: ", cosine)
```

```
similarity:  0.2867214478562801
```

Sentiment analysis

As a next step I am going to conduct a sentiment analysis of the four speeches. I think this is perfect for my case because sentiment analysis is supposed to show if the emotional tone and therefore message is positive or negative. In the context of a political speech, the tone can tell us how that person feels about a certain topic. In terms of an inaugural speech, the tone of the speech can tell

us a lot about what topics will become very important moving forward to the new president, how they feel about it and also what they think about the current state of the country. I also generally expect the tones of speeches to differ between parties (with Republicans potentially more "extreme" in the positives and negatives and Democrats more "balanced").

```
In [74]: files = ["Clinton_Inaug_1993.txt", "GWBush-Inaug-2001.txt", "Obama2009.txt", "Text 3-Trump Inaug.2017.txt"]

         # read lines command

         corpus = []
         for d in files:
             with open( str(d), encoding = 'latin1') as nf:
                 lines = nf.readlines()

             i = 1
             for x in lines:
                 words = x.split()
                 for w in words:
                     corpus.append([d, i, w])

                 i += 1

             #file_path = '/Downloads/'+ str(d)
             #paper_words = pd.DataFrame({'file': [d for d in file_path]})
             #paper_words['text'] = paper_words['file'].apply(lambda x: open(x).read().splitlines())
         corpus
```

```
Out[74]: [['Clinton_Inaug_1993.txt', 1, 'My'],
          ['Clinton_Inaug_1993.txt', 1, 'fellow'],
          ['Clinton_Inaug_1993.txt', 1, 'citizens,'],
          ['Clinton_Inaug_1993.txt', 1, 'today'],
          ['Clinton_Inaug_1993.txt', 1, 'we'],
          ['Clinton_Inaug_1993.txt', 1, 'celebrate'],
          ['Clinton_Inaug_1993.txt', 1, 'the'],
          ['Clinton_Inaug_1993.txt', 1, 'mystery'],
          ['Clinton_Inaug_1993.txt', 1, 'of'],
          ['Clinton_Inaug_1993.txt', 1, 'American'],
          ['Clinton_Inaug_1993.txt', 1, 'renewal.'],
          ['Clinton_Inaug_1993.txt', 1, 'This'],
          ['Clinton_Inaug_1993.txt', 1, 'ceremony'],
          ['Clinton_Inaug_1993.txt', 1, 'is'],
          ['Clinton_Inaug_1993.txt', 1, 'held'],
          ['Clinton_Inaug_1993.txt', 1, 'in'],
          ['Clinton_Inaug_1993.txt', 1, 'the'],
          ['Clinton_Inaug_1993.txt', 1, 'depth'],
          ['Clinton_Inaug_1993.txt', 1, 'of'],
```

```
In [73]: stop_words = list(stopwords.words('english'))
         stop_words
```

```
Out[73]: ['i',
          'me',
          'my',
          'myself',
          'we',
          'our',
          'ours',
          'ourselves',
          'you',
          "you're",
          "you've",
          "you'll",
          "you'd",
          'your',
          'yours',
          'yourself',
          'yourselves',
          'he',
          'him',
```

```
In [56]: for word in corpus: # iterate over word_list
           if word[2] in stop_words:
             corpus.remove(word) # remove word from paperwords if it is a stopword
```

```
In [57]: len(corpus)
```

```
Out[57]: 4676
```

```
Corpus = pd.DataFrame(corpus, columns = ['speech', 'line', 'word'])
Corpus
```

	speech	line	word
0	Clinton_Inaug_1993.txt	1	My
1	Clinton_Inaug_1993.txt	1	fellow
2	Clinton_Inaug_1993.txt	1	citizens,
3	Clinton_Inaug_1993.txt	1	today
4	Clinton_Inaug_1993.txt	1	celebrate
...	...	...	...
4671	Text 3-Trump Inaug.2017.txt	75	You,
4672	Text 3-Trump Inaug.2017.txt	75	And
4673	Text 3-Trump Inaug.2017.txt	75	God
4674	Text 3-Trump Inaug.2017.txt	75	Bless
4675	Text 3-Trump Inaug.2017.txt	75	America.

4676 rows × 3 columns

```
Corpus.shape
```

```
(4676, 3)
```

```
Corpus["word"] = Corpus['word'].str.replace('[^\w\s]','', regex='True')
Corpus
```

	speech	line	word
0	Clinton_Inaug_1993.txt	1	My
1	Clinton_Inaug_1993.txt	1	fellow
2	Clinton_Inaug_1993.txt	1	citizens
3	Clinton_Inaug_1993.txt	1	today
4	Clinton_Inaug_1993.txt	1	celebrate
...	...	...	...
4671	Text 3-Trump Inaug.2017.txt	75	You
4672	Text 3-Trump Inaug.2017.txt	75	And
4673	Text 3-Trump Inaug.2017.txt	75	God
4674	Text 3-Trump Inaug.2017.txt	75	Bless
4675	Text 3-Trump Inaug.2017.txt	75	America

4676 rows × 3 columns

After cleaning the text data again and now starting with the actual sentiment analysis, I first obtained the sentiment list from our NLTK module (that our TAs provided) and turned it into a data frame. I then merged the sentiment list with the text data in order to separate positive from negative words.

```
df_bing = pd.read_csv('bing.csv')
df_bing.head()
```

	Unnamed: 0	word	sentiment
0	1	2-faces	negative
1	2	abnormal	negative
2	3	abolish	negative
3	4	abominable	negative
4	5	abominably	negative

```
result = Corpus.merge(df_bing, on = 'word')
result.head(-5)
```

	speech	line	word	Unnamed: 0	sentiment
0	Clinton_Inaug_1993.txt	1	celebrate	802	positive
1	Clinton_Inaug_1993.txt	13	celebrate	802	positive
2	Clinton_Inaug_1993.txt	15	celebrate	802	positive
3	Text 3-Trump Inaug.2017.txt	11	celebrate	802	positive
4	Clinton_Inaug_1993.txt	1	mystery	4184	negative
...	...	...	...	...	...
538	Text 3-Trump Inaug.2017.txt	58	striving	5777	positive
539	Text 3-Trump Inaug.2017.txt	59	complaining	935	negative
540	Text 3-Trump Inaug.2017.txt	63	thrive	6052	positive
541	Text 3-Trump Inaug.2017.txt	64	miseries	4070	negative
542	Text 3-Trump Inaug.2017.txt	65	pride	4718	positive

543 rows × 5 columns

Here I separated the length of the speech into % parts to see how the sentiment scores are being distributed over it by speech individually. Keep in mind that the speeches are of different length.

```
result['sentiment'] = result['sentiment'].replace('positive', 1)
result['sentiment'] = result['sentiment'].replace('negative', -1)
result['percentage'] = round(result['line'] / max(result['line']) *100/5)*5
```

```
c1993 = result[result['speech'] == 'Clinton_Inaug_1993.txt']
b2001 = result.loc[result['speech'] == 'GWBush-Inaug-2001.txt']
o2009 = result.loc[result['speech'] == 'Obama2009.txt']
t2017 = result.loc[result['speech'] == 'Text 3-Trump Inaug.2017.txt']
```

```
c1993a = c1993[['percentage', 'speech', 'sentiment']].groupby('percentage')['sentiment'].sum()
b2001a = b2001[['percentage', 'speech', 'sentiment']].groupby('percentage')['sentiment'].sum()
o2009a = o2009[['percentage', 'speech', 'sentiment']].groupby('percentage')['sentiment'].sum()
t2017a = t2017[['percentage', 'speech', 'sentiment']].groupby('percentage')['sentiment'].sum()
```

```
c1993a
```

percentage	
0.0	3
5.0	3
10.0	1
15.0	15
20.0	10
25.0	5

Name: sentiment, dtype: int64

```
b2001a
```

percentage	
0.0	3
5.0	6
10.0	0
15.0	4
20.0	4
25.0	10
30.0	6
35.0	−2
40.0	−2
45.0	0
50.0	−1
55.0	15
60.0	6
65.0	8
70.0	4

Name: sentiment, dtype: int64

```
o2009a
```

percentage	
0.0	5
5.0	−9
10.0	−5
15.0	12
20.0	3
25.0	17
30.0	5
35.0	3
40.0	9
45.0	0
50.0	5

Name: sentiment, dtype: int64

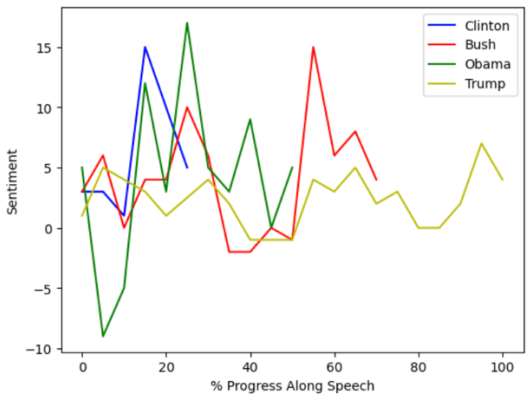
```
t2017a
```

percentage	
0.0	1
5.0	5
15.0	3
20.0	1
30.0	4
35.0	2
40.0	−1
45.0	−1
50.0	−1
55.0	4
60.0	3
65.0	5
70.0	2
75.0	3
80.0	0
85.0	0
90.0	2
95.0	7
100.0	4

Name: sentiment, dtype: int64

Now that the machine has all the data and knows which words to count as negative words and which as positive ones, it shows us a really nice sentiment analysis that is further visualized by the graph below:

```
from numpy import *
import math
import matplotlib.pyplot as plt

plt.plot(c1993a, 'b', label = 'Clinton')
plt.plot(b2001a, 'r', label = 'Bush')
plt.plot(o2009a, 'g', label = 'Obama')
plt.plot(t2017a, 'y', label = 'Trump')
plt.xlabel('% Progress Along Speech')
plt.ylabel('Sentiment')


plt.legend()
plt.show()
```



And again, my theory was entirely wrong: Barack Obama has the most emotionally diverse speech, Bill Clinton has the most positive speech and Donald Trump has the least emotionally diverse speech. G.W. Bush's speech is somewhere between Bill Clinton and Donald Trump. I'm very surprised by this, especially by Donald Trump because I always see him as an extremely emotional speaker. There also again seems to be no similarity between people of the same party which continues to baffle me. Sure, the speeches differ in terms of sentiment, but there are absolute no polar opposites. What makes sense to me though is that because these are all inaugural speeches, they are all generally uplifting/ positive. The only one that starts out with some negative sentiment is Barack Obama's speech but that quickly changes throughout his speech and most of it is very positive.

Wordclouds

Word clouds are a helpful way to visualize how often certain words appear in each speech. The more a word is used, the bigger it shows up in the cloud. Looking at the four presidents, there isn't a ton of overlap in the language they use, especially when comparing across party lines. Clinton and Obama tend to focus more on ideas like service, community, and work, while Bush and Trump lean heavily into national values and strength. That said, some themes do repeat, like the emphasis on "america" and "nation," which stand out clearly in both Bush's and Trump's speeches.

Bill Clinton 1993

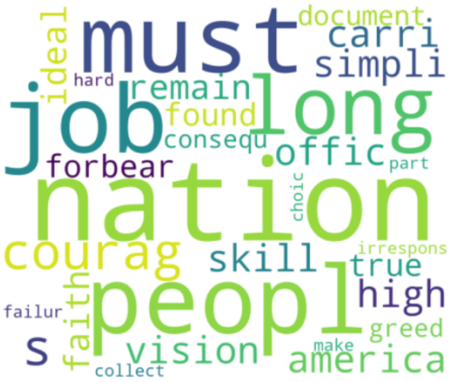
```
Clinton = np.array(Clinton_Speech_BOW)
Clinton = Clinton.tolist()

#Generating a word cloud
from wordcloud import WordCloud, STOPWORDS
cloud_string = ''
for word_list in Clinton:
    cloud_string = cloud_string + ' ' + (' '.join(word_list))
wordcloud = WordCloud(stopwords=STOPWORDS,
                      max_words = 30,
                      background_color='white',
                      width=1200,
                      height=1000).generate(cloud_string)

plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

G. W. Bush 2001

```
Bush = np.array(Bush_Speech_BOW)
Bush = Bush.tolist()

#Generating a word cloud
from wordcloud import WordCloud, STOPWORDS
cloud_string = ''
for word_list in Bush:
    cloud_string = cloud_string + ' ' + (' '.join(word_list))
wordcloud = WordCloud(stopwords=STOPWORDS,
                      max_words = 30,
                      background_color='white',
                      width=1200,
                      height=1000).generate(cloud_string)

plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

Obama 2009

```
Obama = np.array(Obama_Speech_BOW)
Obama = Obama.tolist()

#Generating a word cloud
from wordcloud import WordCloud, STOPWORDS
cloud_string = ''
for word_list in Obama:
    cloud_string = cloud_string + ' ' + (' '.join(word_list))
wordcloud = WordCloud(stopwords=STOPWORDS,
                      max_words = 30,
                      background_color='white',
                      width=1200,
                      height=1000).generate(cloud_string)

plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```



Donald Trump 2017

```
Trump = np.array(Trump_Speech_BOW)
Trump = Trump.tolist()

#Generating a word cloud
from wordcloud import WordCloud, STOPWORDS
cloud_string = ''
for word_list in Trump:
    cloud_string = cloud_string + ' ' + (' '.join(word_list))
wordcloud = WordCloud(stopwords=STOPWORDS,
                      max_words = 30,
                      background_color='white',
                      width=1200,
                      height=1000).generate(cloud_string)

plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```