COGS 108 Final Project

Overview

We have conducted an analysis of lyrics from songs in the U.S. Billboard Top 100 over a timespan of 50 years dating within the years of 1965-2015. After data cleaning we were left with 4,821 songs to analyze after cleaning, which we were able to examine through visualization, grouped visualization, and a permutation test to compare the distributions of songs that we deem as politically motivated between decades. We failed to reject the null hypothesis in our permutation test revealing that there is not enough statistical evidence to imply a change in the distribution of songs containing our political words across decades.

Names and IDs

Joshua Castro - *A13575128*
Cydney Moyle - *A14605173*
Grant Rogers - *A12647853*
Ashley Ruiz - *A13458629*

Research Question

Is there an increase in the amount of politically charged lyrics over time? To answer this question, we are measuring the numerical amount of politically charged lyrics/songs per year in songs record in the Billboard Top 100 from 1965 to 2015 as determined by our lexicon, as well as the amount of non-politically charged lyrics/songs per year, with the aim to draw a comparison between the two.

Background and Prior Work

Recently, there has been a noticeable increase in political activism trending throughout the United States fortified by the use of technology and social media. As technology continues to advance, some forms of communication become more accessible to various communities around the globe thus expanding the network of communication and the ability to share political messages and documentation of oppression and activism. Because of this rise in activism and visibility, we hope to see a correlation between politically charged lyrics and the significance of the year in which they appear. Throughout our analysis of song lyrics we will be using the terms "political" or "political activism" in reference to music that critiques the hegemonic ideologies and policies that systematically work to oppress and exploit and in its efforts functions to promote or impede political reform through messages of resistance.

In “What has America been singing about? Trends in themes in the U.S. top-40 songs: 1960–2010,” Peter G. Christenson et al. analyzed 1,040 U.S. top-40 songs from the years 1960 to 2010 using R software to detect trends in lyrics to which they categorized into 19 themes. While their findings revealed fluctuation and continuity in trends throughout the 5 decades, the most prominent themes of the 19 were "relationships/love" and "sex/sexual desire" (Christenson et al. 2019). On average, the percentage of songs with references to the theme Christenson et al. describe as "social/political issues" was 7.2%. This study confines the "social/political issues" theme to a rap/hip-hop genre because of the genre's background of promoting self-liberation and emancipation through political and social critiques. In a way this limits the theme because the researchers are assuming that theme may only be produced by one sole genre that they describe as having a more salient correlation to themes involving “sex/sexual desire” and “violence”. Furthermore, this research took a look at various themes dating from 1960 to 2010 while ours will take a more in depth analysis of one theme in particular over a more inclusive timeline.

Before the use of R software in 2019, there was another software available called “MusicMetric”, mentioned in “Music Metric Launches Music Industry Trend-Analysis Software”, which composes data on real-time trends arising within the music industry and analyzes the lyrical content while tracking the success or decline of the individual band and record label. This data allows for consumers to then calculate trends, devise opinions, and even determine various aspects of popularity (Routenote.com, 2009).

With these two sources, we are able to continue with our analysis, as we have software equivalently useful to that which is described by Music Metric and we will expanding upon the foundations established by Christenson et al.

Sources

1. What Has America Been Singing About? (https://journals.sagepub.com/doi/full/10.1177/0305735617748205)
2. Music Metric (http://routenote.com/blog/music-metric-launches-music-industry-trend-analysis-software/?fbclid=IwAR0wbGBBsHZ3EJm_5u_VndLrYchSBAJVySmg0B05MmhWCpbBxHLHk7wrAIc)

Hypothesis

We propose that there is a relevant relationship between the songs with lyrics pertaining to a political theme and the year that the song was made popular. Thus, we hypothesize that the amount of politically motivated songs changes with statistical significance across decades.

Dataset(s)

Our dataset is obtained from Kaggle, entitled Billboard 1964-2015 Song Lyrics (https://www.kaggle.com/rakannimer/billboard-lyrics). We found that the raw dataset has 5100 observations, with variables denoting the rank of the song on the Billboard top 100, the artist of the song, the year the song was featured on Billboard, the lyrics of the song, and the source of the song. The data set created by Peter G. Christenson et al. was scraped from each Wikipedia entry of artists and songs using XML and RCurl packages. Similarly, our data set also features thousands of entries across the time span of 50 years, providing data for variables such as song name, artist name, lyrics, time (year), and the ranking of year. TwentyFUN! will be interested in digging deeper into the variables of lyrics and ranking since we assume that the value expressed in the lyrics of the most trending songs during a certain time period will reflect the social values and status.

It is worth mentioning that the title of the dataset entails data inclusive of the year 1964. After working with the data we discovered that this is not the case, and the years start at 1965.

Setup

To help with our analysis, we import the following libraries:

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Data Cleaning

We begin by importing our dataset and examining its characteristics as follows:

```
In [2]: #when importing, use encoding = "ISO-8859-1" to ensure that
        #Unicode characters are converted to ASCII
        data = pd.read_csv("billboard_lyrics_1964-2015.csv",
                           encoding = "ISO-8859-1")
        curr_shape = data.shape
        print("current shape: " + str(curr_shape))
        data.head()
```

```
current shape: (5100, 6)
```

Out[2]:

	Rank	Song	Artist	Year	Lyrics	Source
0	1	wooly bully	sam the sham and the pharaohs	1965	sam the sham miscellaneous wooly bully wooly b...	3.0
1	2	i cant help myself sugar pie honey bunch	four tops	1965	sugar pie honey bunch you know that i love yo...	1.0
2	3	i cant get no satisfaction	the rolling stones	1965		1.0
3	4	you were on my mind	we five	1965	when i woke up this morning you were on my mi...	1.0
4	5	youve lost that lovin feelin	the righteous brothers	1965	you never close your eyes anymore when i kiss...	1.0

```
In [3]: #check which columns have missing values
        num_null = data.isnull().sum()
        num_null
```

```
Out[3]: Rank        0
        Song        0
        Artist      0
        Year        0
        Lyrics    187
        Source    187
        dtype: int64
```

We want to the find observations that are missing entries in the *Lyrics* column. Missing entries may take the form of null values, or are simply blank cells as we can see in row index *2* of the dataframe above. With this, we find null value counts as follows:

In [4]:
```
#establish the condition in which "Lyrics" entries are deemed as missing
#we use np.logical_or to ensure that we are getting NaN or '   '
missing_entry = np.logical_or(data["Lyrics"].isnull(),
                              data["Lyrics"] == '   ')
#get years with null values
years = data.loc[missing_entry].groupby("Year").count()
#get the counts of years with null values
sorted_years = years.sort_values(by = ['Rank'], ascending = False)
sorted_years = sorted_years.rename(index = str,
                                   columns =
                                   {'Rank':'Null Count'})[['Null Count']]
sorted_years.T
```

Out[4]:

Year	1971	1972	1969	1996	1977	1981	1968	1997	1974	1966	...	2001	1983	1987	1993
Null Count	13	12	10	9	9	8	8	8	8	8	...	2	2	2	2

1 rows × 49 columns

We find that null values are fairly spread out amongst the *Year* column. We decided that the integrity of our analysis would remain intact in dropping the null values, as there is no way to save them since our analysis depends on the lyrics of the songs. Since no year hordes most the null values, there will be enough observations per year to ensure that our analysis remains trustworthy. However, we will hold the songs with missing lyrics for possible future analysis

In [5]:
```
#get the indices with missing entries
missing_indices = data.loc[missing_entry].index.tolist()
#hold the songs with missing entries for possible further analysis.
missing_entries = data.loc[missing_entry]
#drop rows according to missing_indices
no_null = data.copy().drop(index = missing_indices)
#get the new shape to verify rows with null entries have been dropped
new_shape = no_null.shape
print("New shape: " + str(new_shape))
no_null.head()
```

```
New shape: (4868, 6)
```

Out[5]:

	Rank	Song	Artist	Year	Lyrics	Source
0	1	wooly bully	sam the sham and the pharaohs	1965	sam the sham miscellaneous wooly bully wooly b...	3.0
1	2	i cant help myself sugar pie honey bunch	four tops	1965	sugar pie honey bunch you know that i love yo...	1.0
3	4	you were on my mind	we five	1965	when i woke up this morning you were on my mi...	1.0
4	5	youve lost that lovin feelin	the righteous brothers	1965	you never close your eyes anymore when i kiss...	1.0
5	6	downtown	petula clark	1965	when youre alone and life is making you lonel...	1.0

We also recognize that we are not interested in the *Source*, *Song*, *Rank*, or *Artist*, as we are determining the trend of politically charged lyricism over time. So, we continue with only the *Year* and *Lyrics* columns in our dataframe.

In [6]:
```
#drop the 'Rank', 'Song', 'Artist', and 'Source' columns
df = no_null[['Year', 'Lyrics']]
#get the final shaped to verify that columns have been dropped
final_shape = df.shape
print("Final shape: " + str(final_shape))
df.head()
```

```
Final shape: (4868, 2)
```

Out[6]:

	Year	Lyrics
0	1965	sam the sham miscellaneous wooly bully wooly b...
1	1965	sugar pie honey bunch you know that i love yo...
3	1965	when i woke up this morning you were on my mi...
4	1965	you never close your eyes anymore when i kiss...
5	1965	when youre alone and life is making you lonel...

Now that we have established how our dataframe and we have removed missing entries, we tokeniz the entries of *Lyrics* by word so that it would be easier to compare lyrics to our lexicon.

In [7]:
```python
## import word tokenizer
from nltk.tokenize import word_tokenize

#make a copy of df to perform split on
temp = df.copy()
#tokenize the words in lyrics
temp['Lyrics'] = temp['Lyrics'].apply(word_tokenize)
df = temp
#check the new dataframe
df.head()
```

Out[7]:

	Year	Lyrics
0	1965	[sam, the, sham, miscellaneous, wooly, bully, ...
1	1965	[sugar, pie, honey, bunch, you, know, that, i,...
3	1965	[when, i, woke, up, this, morning, you, were, ...
4	1965	[you, never, close, your, eyes, anymore, when,...
5	1965	[when, youre, alone, and, life, is, making, yo...

Data Analysis & Results

To first gain an understanding of our data set, we plan on getting a baseline understanding of what words are most unique amongst all the songs, then count the number of occurrences of political words in each song. Once we have a better understanding of our dataset, we will hone in our analysis to see if our hypothesis stands true.

Let us begin with TF/IDF. We first clean out the stopwords from our Lyrics column so we can find the more significant words in each song. We define a helper method to filter our the stop words as follows:

In [8]:
```
# import stop words
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))

#helper method to filter songs
def filter_sw(song):
    #list to hold all the filtered words
    filtered = []
    #if song is not a stop word append to a list
    for word in song:
        if word not in stop_words:
            filtered.append(word)
    return filtered

#apply the helper method to our dataframe for a new column
df['Relevant_Words'] = df['Lyrics'].apply(filter_sw)
df.head()
```

Out[8]:

	Year	Lyrics	Relevant_Words
0	1965	[sam, the, sham, miscellaneous, wooly, bully, ...	[sam, sham, miscellaneous, wooly, bully, wooly...
1	1965	[sugar, pie, honey, bunch, you, know, that, i,...	[sugar, pie, honey, bunch, know, love, cant, h...
3	1965	[when, i, woke, up, this, morning, you, were, ...	[woke, morning, mind, mind, got, troubles, who...
4	1965	[you, never, close, your, eyes, anymore, when,...	[never, close, eyes, anymore, kiss, lips, ther...
5	1965	[when, youre, alone, and, life, is, making, yo...	[youre, alone, life, making, lonely, always, g...

We now perform TF/IDF on the relevant words. We set up a vectorizer to perform our analysis. We also want to make sure our relevant lyrics are in a format the vectorizer could transform. With this, we will then find the most important word over all the songs.

In [9]:
```python
#import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

#initialize vectorizer
tfidf = TfidfVectorizer(sublinear_tf=True,
                        analyzer='word',
                        max_features = 2000,
                        stop_words = stop_words)

#put the relevant words into a format the vectorizer could work with
relevant_words = [str(list(df.Relevant_Words))]

#transform the relevant words with the vectorizer
lyrics_tfidf = pd.DataFrame(
    tfidf.fit_transform(relevant_words).toarray())
#rename the columns so we know the tf/idf for each word
lyrics_tfidf.columns = tfidf.get_feature_names()
#get the most important word of all the words throughout all the songs
lyrics_tfidf.idxmax(axis = 1)
```

Out[9]:
```
0    love
dtype: object
```

Aw, how cute! ***Love*** is the most important word over all the songs on the Billboard Top 100 from 1965 - 2015, according to our TF/IDF vectorizer. To delve deeper, we want to see what the most relevant words are of the given songs per decade to see how the general dynamic of songs changes. To do this, we group the words by ten year increments, and then we perform TF/IDF on the relevant words of that decade.

In [10]:
```python
#first decade to check
decade = [1965, 1975]

#only go up to 2005 - 2015 decade
while decade[1] <= 2015:
    #get the relevant words of the data within the described decade
    by_decade = df.loc[np.logical_and(df['Year'] >= decade[0],
                                      df['Year'] <= decade[1])]
    relevant_words = [str(list(by_decade.Relevant_Words))]

    #perform TF/IDF on the relevant words
    decade_tfidf = pd.DataFrame(tfidf.fit_transform(relevant_words).toarray
    #rename the columns so we know the tf/idf for each word
    decade_tfidf.columns = tfidf.get_feature_names()
    #print the results per decade
    display('decade: ' + str(decade) + '; TF/IDF results: '
            + decade_tfidf.idxmax(axis = 1))

    #increment the decade to the next decade
    decade[0] += 10
    decade[1] += 10
```

```
0    decade: [1965, 1975]; TF/IDF results: love
dtype: object

0    decade: [1975, 1985]; TF/IDF results: love
dtype: object

0    decade: [1985, 1995]; TF/IDF results: love
dtype: object

0    decade: [1995, 2005]; TF/IDF results: im
dtype: object

0    decade: [2005, 2015]; TF/IDF results: im
dtype: object
```

Interesting results. We see here that the most important word in the Billboard Top 100 changes at around 1995, which suggests that dynamic of what was conceived as popular could have shifted around the mid 90's. This motivates our analysis to look for any sort of shift in usage of politically-charged words in songs at around the mid-90's.

To continue, we define a list of words with possibly political charge behind it. For transparency, by politically charged we imply that the list of words has had historical affiliation with politics. We looked to recent events with possibly political motivations or reactions to political action to inspire the words in our list. We understand that this list is not the end-all-be-all list of political words in songs, but we would like to examine how the songs in this dataset may possibly relate to our chosen words.

Moving forward, we would like to apologize to anyone how sees these words as ethically incorrect to include within our lexicon. We have spent time deliberating on what words, names, and events could have inspired political action within recent years, ultimately deciding to include names and events keeping in mind the possible trauma coming with these names and events. Again, we continue with our analysis using these words, but would like to apologize to those who disagree with our choice of words.

```
In [11]: pol_words = ["administration", "government", "public", "police",
                      "political", 'politics', "state", "representation",
                      "jim", "crow", "civil", "rights", "freedom",
                      "rodney","agenda","mlk", "boycott", "reagan",
                      "bush", "nixon", "clinton", "obama", "trayvon",
                      "martin", "war", "vietnam", "afghanistan", "iraq",
                      "refugee", "jail", "columbine", "terrorist",
                      'terrorism',"america", "u.s.a." "usa" "congress",
                      "president", "vote", "9/11", "economy", "tax",
                      "chapo","american", "country", "nation", 'revolt',
                      'revolution', 'protest', 'gun', 'colonizer', 'colonialism',
                      'bomb', 'shoot', 'fbi', 'cia', 'kkk',
                      'ira', 'exploit', 'exploitation', 'mandela', 'military',
                      'republican', 'democrat', 'rochester', 'riot',
                      'riots', 'fascist']
```

With this "lexicon" of words we seem to have a connection with recent politically-charged events, we now examine the frequency of these words in our dataset of songs. We first write a helper method for counting the number of times a word in a song appears in our lexicon. Moving forward, we are installing into our dataframe columns that we could use to further our analysis. By adding the number of political words and the words which appear as an empirical counter and a unique counter, we will be able to analyze how impactful these words are in the song featuring them.

In [12]:
```
def political_word_counter(song):
    #initialize counters
    count = 0
    unique_count = 0
    #intialize list to hold words that appear in songs
    found = []
    #list of unique words found
    unique_found = []
    #check each if a word in our lexicon appears in the song
    for word in pol_words:
        #get the number of times the political word is said
        for lyric in song:
            if lyric == word:
                count += 1
                found.append(word)
                #we also want to hold the number of unique political words
                if word not in unique_found:
                    unique_found.append(word)
                    unique_count += 1
    return count, found, unique_found, unique_count

#apply the helper method to the Lyrics column
df['Political_Ct'] = df['Lyrics'].apply(lambda x:
                                        political_word_counter(x)[0])
df['Political_Words'] = df['Lyrics'].apply(lambda x:
                                           political_word_counter(x)[1])
df['Unique_Political_Words'] = df['Lyrics'].apply(lambda x:
                                                  political_word_counter(x)
df['Unique_Political_Ct'] = df['Lyrics'].apply(lambda x:
                                               political_word_counter(x)[3]
#peek results
df.head()
```

Out[12]:

	Year	Lyrics	Relevant_Words	Political_Ct	Political_Words	Unique_Political_Words	Unique
0	1965	[sam, the, sham, miscellaneous, wooly, bully, ...	[sam, sham, miscellaneous, wooly, bully, wooly...	0	[]	[]	
1	1965	[sugar, pie, honey, bunch, you, know, that, i,...	[sugar, pie, honey, bunch, know, love, cant, h...	0	[]	[]	
3	1965	[when, i, woke, up, this, morning, you, were, ...	[woke, morning, mind, mind, got, troubles, who...	0	[]	[]	
4	1965	[you, never, close, your, eyes, anymore, when,...	[never, close, eyes, anymore, kiss, lips, ther...	0	[]	[]	
5	1965	[when, youre, alone, and, life, is, making, yo...	[youre, alone, life, making, lonely, always, g...	0	[]	[]	

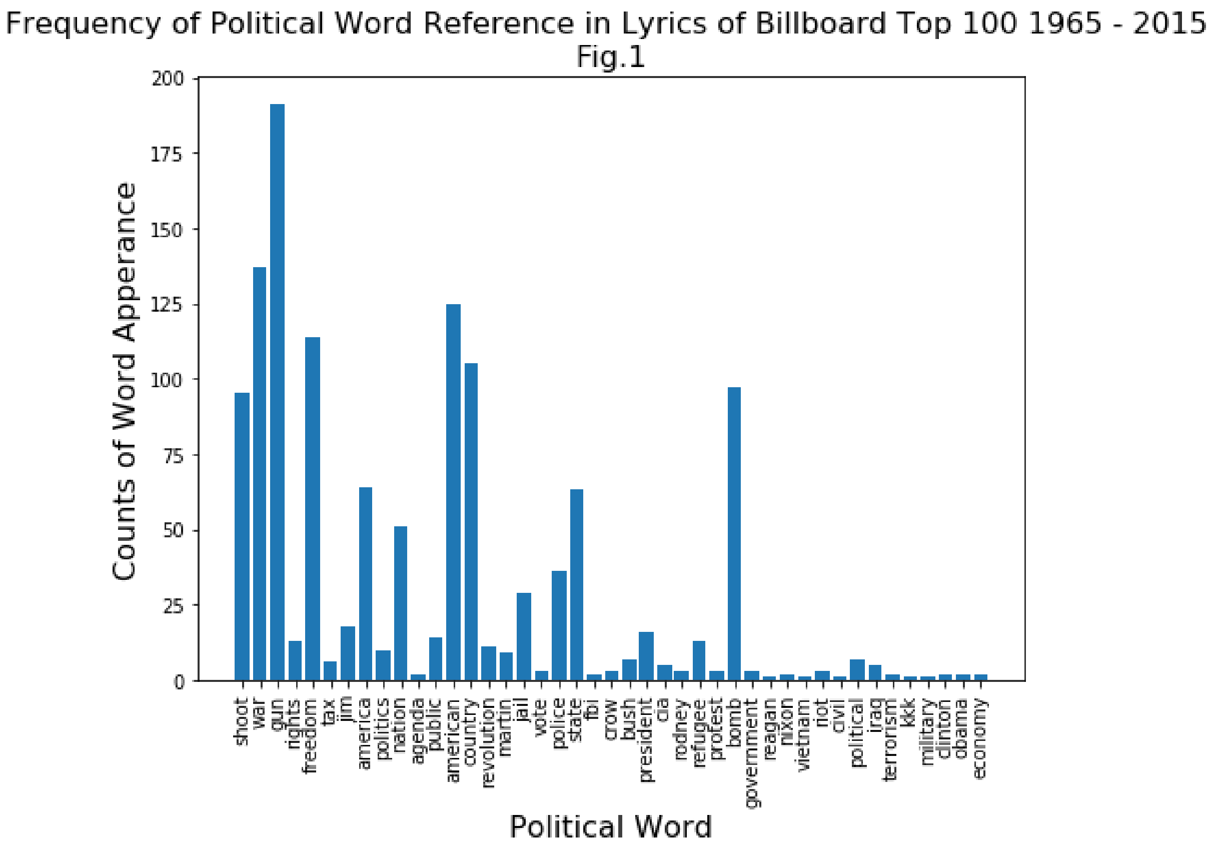
With a new reference of what political words from our lexicon were used throughout our dataset in the *Political_Words* column, we plot out some quick graphs to see what we can learn about how our political words are used throughout the Billboard Top 100 from 1965 to 2015. We start by working toward making a histogram to show the frequency of each political word referenced in the dataset.

```
In [13]: #dictionary to hold the counts of each used political word
record = {}

#helper method to count the used political words
def frequency_helper(words, record):
    #only take words in the song that are featured in our lexicon
    for word in words:
        if word in pol_words:
            #increment count in dictionary if exists
            if word in record:
                record[word] += 1
            #initialize count in dictionary if not found
            else:
                record[word] = 1

#apply the helper function to get the frequencies of our words
df.Political_Words.apply(lambda x: frequency_helper(x, record))

#plot the findings
plt.figure(1,figsize = (8, 6))
plt.bar(record.keys(), record.values())
plt.xticks(rotation=90)
plt.title('Frequency of Political Word Reference in Lyrics '
          +'of Billboard Top 100 1965 - 2015 \nFig.1',
          fontsize = 16)
plt.ylabel('Counts of Word Apperance', fontsize = 16)
plt.xlabel('Political Word', fontsize = 16);
```



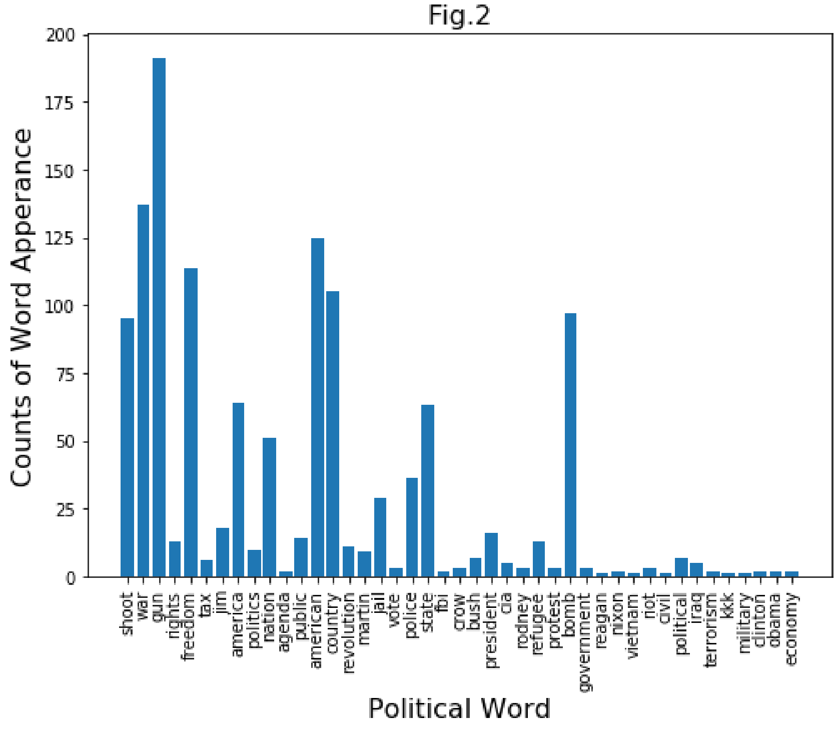
We see that one of our more frequently used political words in 'gun'. The word 'gun' may or may not have political charge to it, so we choose to keep songs which reference the word 'gun' along with another word in our lexicon. We do this because we believe that if the word 'gun' is referenced in a song with other political words referenced, the song is more likely to have political motivations.

To accomplish this, we look to the drop the instances in which the word 'gun' is the only unique political word. We first look at the frequency of unique political words per song in our dataframe.

In [14]:
```
#dictionary to hold the counts of each used political word
unique_record = {}
#apply the helper function to get the frequencies of our words
df.Political_Words.apply(lambda x: frequency_helper(x, unique_record))

#plot the updated findings
plt.figure(2,figsize = (8, 6))
plt.bar(unique_record.keys(), unique_record.values())
plt.xticks(rotation=90)
plt.title('Updated Frequency of Political Word Reference in Lyrics'
          +' of Billboard Top 100 1965 - 2015 \nFig.2',
          fontsize = 16)
plt.ylabel('Counts of Word Apperance', fontsize = 16)
plt.xlabel('Political Word', fontsize = 16);
```

Updated Frequency of Political Word Reference in Lyrics of Billboard Top 100 1965 - 2015
Fig.2



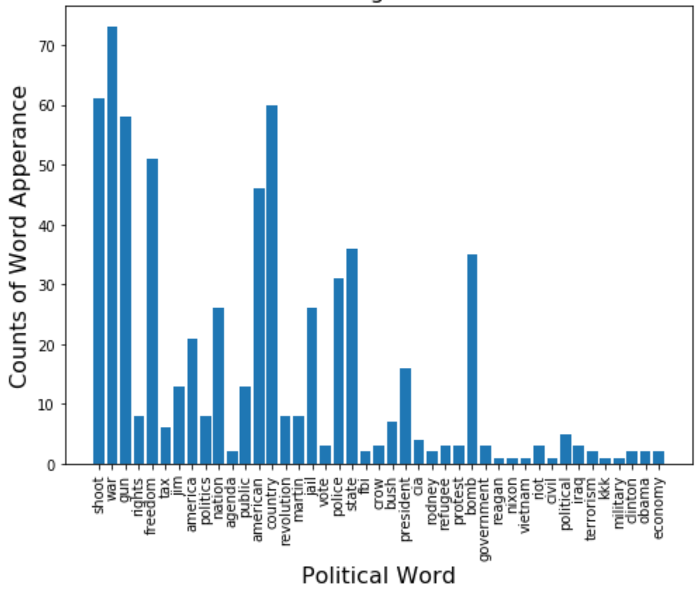
From our original dataframe, we confirm that *gun* is the most used unique political word per song. We perform our cleaning of songs that only have 'gun' as the unique political word as follows:

```
In [15]: #get index of songs that only feature 'gun'
         gun_songs = df.loc[df.Unique_Political_Words.apply(lambda x: 'gun' in x)]
         to_drop = gun_songs.loc[df.Political_Ct  == 1].index

         #drop the songs according to our found indices
         df.drop(to_drop, inplace = True)
         #dictionary to hold the counts of each used political word
         updated_unique_record = {}
         #apply the helper function to get the frequencies of our words
         df.Unique_Political_Words.apply(lambda x:
                                         frequency_helper(x, updated_unique_record))

         #plot the updated findings
         plt.figure(3,figsize = (8, 6))
         plt.bar(updated_unique_record.keys(), updated_unique_record.values())
         plt.xticks(rotation=90)
         plt.title('Frequency of Unique Political Word References '
                   +'amongst all Unique'
                   +' Lyrics of Billboard Top 100 1965 - 2015 \nFig.3',
                   fontsize = 16)
         plt.ylabel('Counts of Word Apperance', fontsize = 16)
         plt.xlabel('Political Word', fontsize = 16);
```

Frequency of Unique Political Word References amongst all Unique Lyrics of Billboard Top 100 1965 - 2015
Fig.3

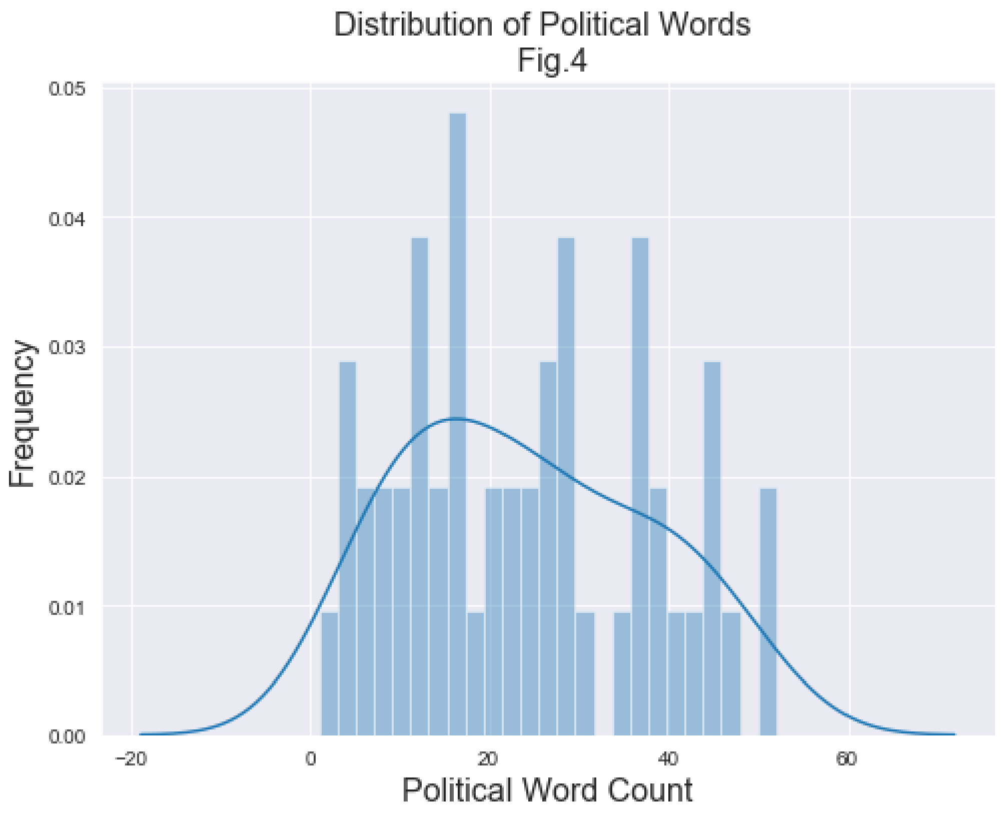


We now see that the word 'war' is the most frequently used unique political word per song amongst our described political words. We move forward with these update findings as the most-frequently used words have more believable possible political motivations. We will keep in mind that we cannot guarantee that the dropped songs did not have political-charge, and will continue our analysis as follows.

We are also interested in the distribution of political word count of songs per year. Let's explore that below!

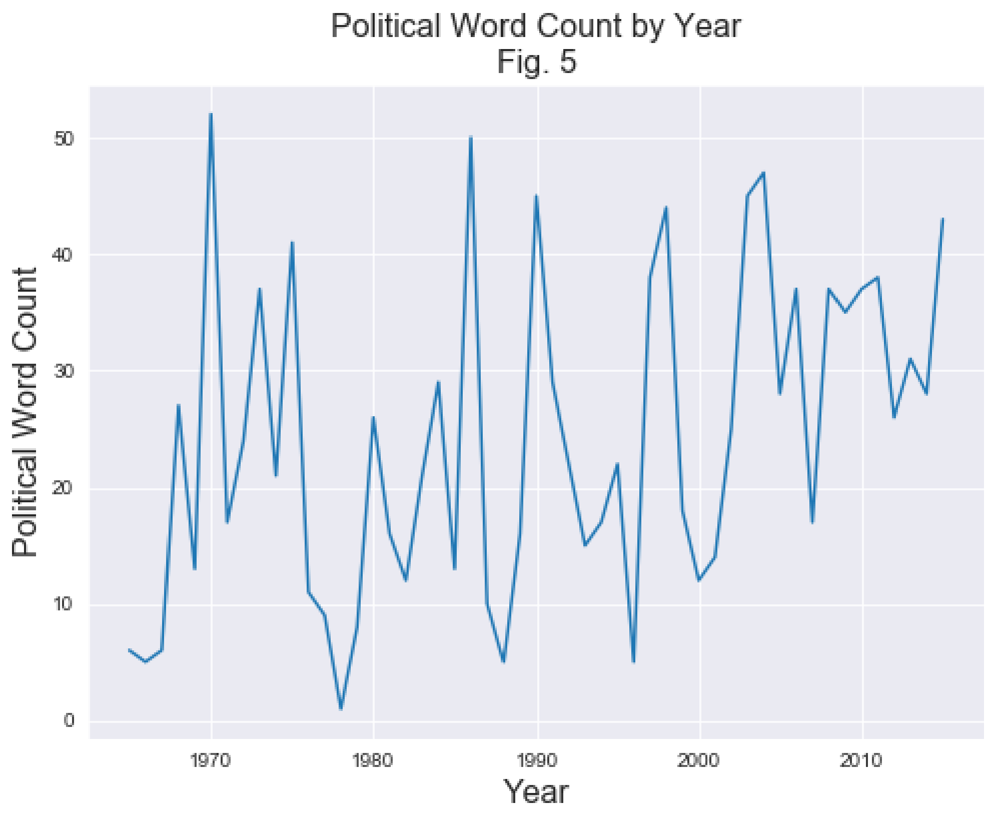
```
In [23]: #get the sum of political word counts by year to put into line graph
         political_counts_by_yr = df.groupby("Year")['Political_Ct'].sum()

         sns.set_style('darkgrid');
         plt.figure(4, figsize = (8, 6));
         fig4 = sns.distplot(political_counts_by_yr, bins = 25);
         fig4.set_title('Distribution of Political Words \n Fig.4', fontsize = 16);
         fig4.set_ylabel('Frequency', fontsize = 16);
         fig4.set_xlabel('Political Word Count', fontsize = 16);
```



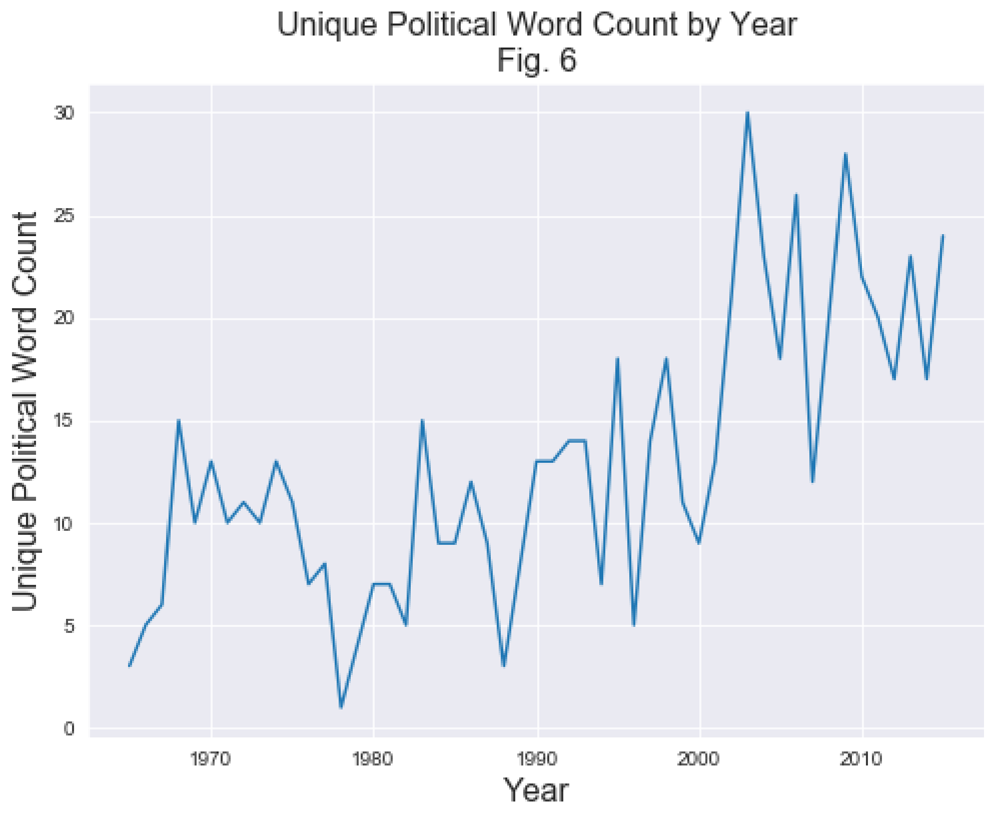
From the graph, we can see that the distribution is slightly skewed to the right and slightly bimodal. There is not much we can do with this information to further our analysis, so we will use lineplot to visualize trends over time.

In [17]:
```
#plot the findings
plt.figure(5,figsize = (8, 6))
political_counts_by_yr.plot(kind = 'line')
plt.title('Political Word Count by Year\nFig. 5', fontsize = 16)
plt.ylabel('Political Word Count', fontsize = 16)
plt.xlabel('Year', fontsize = 16);
```



```
In [18]: #get the sum of political word counts by year to put into line graph
         un_political_counts_by_yr = df.groupby("Year")['Unique_Political_Ct'].sum()

         #plot the findings
         plt.figure(6,figsize = (8, 6))
         un_political_counts_by_yr.plot(kind = 'line')
         plt.title('Unique Political Word Count by Year\nFig. 6', fontsize = 16)
         plt.ylabel('Unique Political Word Count', fontsize = 16)
         plt.xlabel('Year', fontsize = 16);
```



In Fig. 5, we see that there is high variability in our lineplot from the year 1965 up to around 1995, but at 1995 the variance decreases and there is some sort of linear growth. We also see this pattern in Fig. 6, as though the total number of our political words is used throughout the years remains relatively low, we still see some sort of positive growth in unique political word usage from the year 1990 throughout the 2000's. This matches our findings with the TF/IDF lyric analysis by decade, reinforcing the idea there was some sort of change in the general dynamic of songs featured on the Billboard Top 100 from 1995 and onwards. With this, we continue our analysis keeping this initial visualization in mind, and seeing if there is any sort of further relationship that we can establish.

To wrap up our analysis, we want to perform a permutation test to see if there is a statistically significant difference between the distribution of songs featuring political words and songs not featuring political words per year between decades. The purpose of our permutation test is to see if shuffling the classifications of songs featuring our political words and songs not featuring our political words can check whether or not this difference in distributions is due to random chance or not. Essentially, we verify that these distributions are meant to be per year, allowing us to make a conclusion about our hypothesis.

For each year, we define the following hypotheses:

1. *Null hypothesis H_0*: There is no statistically significant difference in the distribution of songs featuring our chosen political words and the distribution of songs not featuring our chosen poltical words amongst the Billboard Top 100 songs between the two described decades.
2. *Alternative hypothesis H_1*: There is a statistically significant difference in the our two described distributions between the two described decades.

We choose a significance level of $\alpha = 0.05$. Our test statistic is the total variation difference between the distributions of songs featuring our chosen political words and songs not featuring our chosen political words. Our described decades are (1965, 1975), (1975, 1985), (1995, 2005), (2005, 2015). We continue as follows:

```
In [19]: #map whether songs contain political words or not to True or False
df['is_political'] = df['Political_Ct'].apply(lambda x: x > 0)

#helper method to map each year to a decade
def map_decades(year):
    #list out decades
    decades = [(1965, 1975), (1976, 1985),(1986, 1995),
               (1996, 2005),(2006, 2015)]
    #find which decade the year falls in
    if year >= 1965 and year <= 1975:
        return decades[0]
    elif year >= 1976 and year <= 1985:
        return decades[1]
    elif year >= 1986 and year <= 1995:
        return decades[2]
    elif year >= 1996 and year <= 2005:
        return decades[3]
    else:
        return decades[4]

#apply helper method to map year to decade
df['decade'] = df['Year'].apply(map_decades)
```

We write helper methods for our permutation test:

In [20]:
```python
def simulate_null(curr_year):

    #list to hold tvds
    tvds = []

    for _ in range(500):

        #shuffle the decade column
        shuffled_col = (
            curr_year['decade']
            .sample(replace=False, frac=1)
            .reset_index(drop=True)
        )

        # put them in a table
        shuffled = (
            curr_year
            .assign(**{
                #replace decade with shuffled_col
                'decade': shuffled_col,
                #map if song contains political words to True or False
                'is_political': df['Political_Ct'].apply(lambda x: x > 0)
            })
        )

        # compute the tvd
        shuffled = (
            shuffled
            #pivot table for easier comparison of our findings
            .pivot_table(index='is_political', columns='decade',
                         values = 'Year', aggfunc = 'count')
            .apply(lambda x:x / x.sum(), axis=1)
        )

        #calculate the total variation distance
        tvd = shuffled.diff().iloc[-1].abs().sum() / 2
        tvds.append(tvd)

    return tvds

def permutation_result(curr_year, tvds):
    observed_data = (
            curr_year
            #pivot table for easier comparison of our findings
            .pivot_table(index='is_political', columns='decade',
                         values = 'Year', aggfunc = 'count')
            .apply(lambda x:x / x.sum(), axis=1)
        )

    obs = observed_data.diff().iloc[-1].abs().sum() / 2
    pval = np.mean(tvds > obs)

    if pval > 0.05:
        #fail to reject
        return 'FAIL TO REJECT NULL HYPOTHESIS'
    else:
        return 'REJECT NULL HYPOTHESIS'
```

With these helper methods, we perform the permutation test between pairs of decades as follows:

```
In [21]: #define what decades we are checking

         decades = [(1965, 1975), (1976, 1985),(1986, 1995),
                    (1996, 2005),(2006, 2015)]

         #list to hold total variation distances
         tvds = []

         #perform permutation test
         for decade in decades:

             #stop comparisons once our initial decade is 2015
             if decade == (2006, 2015):
                 break
             next_decade = (decade[1], decade[1]+10)
             #try to run the test
             try:
                 #get the data from our range of years
                 curr_year = df.loc[np.logical_and(df.Year >= decade[0],
                                                   df.Year <= (next_decade[1]))]
                 #simulate the null hypothesis for our test using the helper
                 tvds = simulate_null(curr_year)
             except:
                 pass

             #show results
             display('From '+str(decade)+ ' to ' +str(next_decade)+
                     ' we ' + permutation_result(curr_year, tvds))
```

```
'From (1965, 1975) to (1975, 1985) we FAIL TO REJECT NULL HYPOTHESIS'

'From (1976, 1985) to (1985, 1995) we FAIL TO REJECT NULL HYPOTHESIS'

'From (1986, 1995) to (1995, 2005) we FAIL TO REJECT NULL HYPOTHESIS'

'From (1996, 2005) to (2005, 2015) we FAIL TO REJECT NULL HYPOTHESIS'
```

We see that through all the pairs of decades, we fail to reject the null hypothesis. This means that there is no statistical signficance in the difference in distributions between songs featuring our chosen political words and songs not featuring our chosen political words. Though the results are not that which we would have desired to see, these results makes sense because the number of songs with featuring words from our chosen political words must be miniscule in size compared to the number of songs not featuring our chosen words.

With the results from our permutation test, we state with 95% confidence that there has not been a statistically significant change in the number of songs with political words over time from the data derived from the Billboard Top 100 from 1965 - 2015.

Ethics & Privacy

For our Billboard's pop songs dataset, it is important for us to consider whether the preferences of people from non-white ethnic backgrounds are being equally weighted in determining what songs are deemed as popular for the year or not. Since the Billboard's Top 100 reflects the popular culture of the United States in it's respective contemporary, we must take into account the significant bias due to the discrimination of non-white heterosexual artists predating long before 1964 that is still relevant in the United State's and its various forms of media including Billboard's Top 100.

Another consideration is the possible effect of generational differences in dialect and culture on trends in the lyrics of songs in the U.S. Billboard's Top 100. Considering the fact that our data from the U.S. Billboard's Top 100 encapsulates various generations beginning with the baby boomers, for example, there is reason to believe that representation of ideologies and practices in pop music progresses between generations with the passage of time. We have found that our dataset is already well protected in terms of privacy considering the information in the dataset is all from the U.S. Billboard Top 100, a public source. The only exception to this rule lies in the inclusion of names within the song lyrics. We included these names with the assumption that they were pre-approved by their likeness, considering that (1) they were in published and copyrighted songs and (2) located on a public source. Additionally, there may also be an unknown bias within the Billboard Top 100 algorithm or process in which they establish the Top 100 songs. As data scientists, we cannot trust that the Billboard Top 100 was an unbiased representation of popular music over the totality of the fifty years of collected data. For instance, we know that our dataset contains information as far back as 1964. By the year 1964, the Civil Rights Act was signed, and in the successive year 1965, the Voting Rights Act was passed, yet there is no notable increase in political words in the lyrics of songs.

Furthermore, there is an inherent bias within our lexicon of what we define as political words and how we define what "political" is. This bias is due to the fact that the lexicon was not created or influenced by any particular research. While it is our hope that it is culturally sensitive and captures relevant political lyrics, it is statistically impossible to capture every unique lyric. Additionally, our lexicon may not be fully representative of each and every generation and culture, which is also a significant bias to be considered. To add, in determining the ethicality of our dataset and our analysis, we must look deeper as to what our findings may imply or how our findings may be used for harm and found that Type 1 error could arise from our findings in the misinterpretations of songs, i.e they do not actually promote messages of resistance. Our analysis of missing values revealed that after 1971 there is a gradual decrease in the null values regarding lyrics and we assume that this is due to a poor quality of maintenance and record keeping. We are inferring this because by the year 2000 there is only 1 song that is missing lyrics in contrast to the year 1971 where there were 13 null values. A further examination of this finding may also disclose the dependability and authenticity of the U.S. Billboard Top 100.

Conclusion & Discussion

While we acknowledge that our visualizations and permutation test deny our original hypothesis, we would also like to state that there are fundamental flaws in our chosen dataset and certain liberties taken that prevent us from drawing conclusions on a larger scale. We came to the conclusion that there is not enough statistical evidence to state that there is a difference in the distribution of songs deemed political or not throughout the decades. So, we cannot conclude that with more visibility of recent activist events, songs have become more politically motivated or charged.

Although there is an explicit history of civil activism spanning 1964-2015, the popular music of this time tends to only represent hegemonic themes such as heterosexual love or pastimes. Thus, while the Billboard Top 100 dataset encompasses thousands of songs, it may not be indicative of the sentiment of every single song released per year. Our political word lexicon is not an all encompassing selection of possibly politically motivated words, so we could have missed out on other possibly political words or phrases in songs.

If studies were to continue, recommend that future data scientists replicating this analysis draw from a larger dataset encompassing a wider and more unique variety of songs per year, rather than the most popular, and establish a more inclusive lexicon. Given this fact, it would also be interesting to consider how genre factors into politicized lyrics as well.