# Democratic Debate Transcript Analysis

Currently, I'm studying abroad in Paris taking a French class and a Philosophy course that is interesting but it just doesn't satisfy my need to have data support my claims. As I watched the democratic debates last week, I thought perhaps I could write a program to analyze the candidates comments. I've realized The New York Times and CNN may have already done there own analysis, but I know how to write code in Python and there are several questions I'm interested in exploring that might not have appeard in those company's analysis. Firstly, I copied and pasted the text of the transcripts from Time Magazine and saved them locally in `night1.txt` and `night2.txt` .

In [1]:
```
import pandas as pd, matplotlib.pyplot as plt, numpy as np
import re, string, nltk
from collections import defaultdict
from nltk.corpus import stopwords
```

In [2]:
```
nltk.download("punkt")
```

```
[nltk_data] Downloading package punkt to /Users/dParris/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[2]:
```
True
```

In [3]:
```
def readTranscripts(fileName):
    with open(fileName) as f:
        lines = f.readlines()
    f.close()
    transcript = lines[4:]
    return ''.join(transcript).split("\n")
```

In [4]:
```
night1 = readTranscripts('./night1.txt')
night1[0:5]
```

Out[4]:
```
['    HOLT: Good evening, everyone. I’m Lester Holt, and welcome to the first De
mocratic debate to the 2020 race for president.',
 '',
 '    GUTHRIE: Hi, I’m Savannah Guthrie. And tonight, it’s our first chance to s
ee these candidates go head to head on stage together.',
 '',
 '    GUTHRIE: We’ll be joined in our questioning time by our colleagues, Jose D
iaz-Balart, Chuck Todd, and Rachel Maddow.']
```

In [5]:
```
night2 = readTranscripts('./night2.txt')
night2[0:5]
```

Out[5]:
```
['    HOLT: And good evening once again. Welcome to the candidates and our spiri
ted audience here tonight in the Arsht Center and across America. Tonight we con
tinue the spirited debate about the future of the country, how to tackle our mos
t pressing problems and getting to the heart of the biggest issues in this Democ
ratic primary.',
 '',
 '    GUTHRIE: Tonight we are going to talk about healthcare, immigration. We're
also to dive into the economy, jobs, climate change as well.',
 '',
 '    DIAZ-BALART: As a quick rules of the road before we begin and they may sou
nd familiar 20 candidates cal—qualified for this first debate. As we said we hea
rd from 10 last night and we will hear from 10 more tonight. The breakdown for e
ach night was selected at random. The candidates will have 60 seconds to answer,
30 seconds for follow-ups.']
```

Above is an example of how the data is structured in the .txt file. I've noticed seems that the odd indices of these first few lines are empty strings. If this is invariant, our analysis will be a bit easier. First, let's make sure this is consistent throughout the file for the first night. Running the cell below should print only if there is some non-blank odd line in the file.

In [6]:
```
for i in range(1, len(night1), 2):
    if night1[i] != '':
        print("not blank")
```

In [7]:
```
for i in range(1, len(night2), 2):
    if night2[i] != '':
        print("not blank")
```

Nothing printed, so the odd indices of `night1` and `night2` are all blank lines. Parfait!!

In [8]:
```
night1 = night1[::2]
night2 = night2[::2]
```

Now, the fun begins. We'll make a map from each speaker to a list of their conversations.

In [9]:
```
def isSpeaker(s, speakers):
    return s[-1] == ":" and s[:-1] in speakers
```

In [10]:
```
def analyzeCandidates(lst):
    """
    input: List(str)= the conversations of each night, where each speaker has a co
lon in the last place
    output: dictionary mapping speakers to a list of their inputs during the nigh
t.
    """
    speakers = findSpeakers(lst)
    speakersToText = {}
    for s in speakers:
        speakersToText[s] = []
        #print('\t', s)

    wordByWord = ''.join(lst).split()

    start = 0
    end = 1
    while end < len(wordByWord):
        speaker = wordByWord[start][:-1]
        mightSpeak = wordByWord[end]
        if isSpeaker(mightSpeak, speakers):
            currLst = speakersToText[speaker]
            currLst.append(' '.join(wordByWord[start + 1:end]))
            speakersToText[speaker] = currLst
            start = end
        end += 1
    return speakersToText
```

In [11]:
```
def findSpeakers(lst, deBlasio=False):
    speakers = set()
    for s in lst:
        if ":" in s:
            idx = s.index(":")
            if idx < 20:
                speakers.add(s[:idx].strip())
    #================ helping Mayor De Blasio have a voice ===============#
    if deBlasio:
        speakers.remove("DE BLASIO")
        speakers.add("DEBLASIO")
    #================ helping Mayor De Blasio have a voice ===============#
    return speakers
```

In [12]:
```
candidates1 = analyzeCandidates(night1)
for candidate in candidates1.keys():
    print(candidate, len(candidates1[candidate]))
```

```
TODD 71
GABBARD 15
GUTHRIE 32
(UNKNOWN) 6
DELANEY 28
HOLT 40
BOOKER 18
DE BLASIO 0
INSLEE 9
CASTRO 23
KLOBUCHAR 15
ANNOUNCER 1
MADDOW 36
RYAN 21
DIAZ-BALART 28
O'ROURKE 28
WARREN 16
```

UH OH, apparently Mayor De Blasio didn't make any inputs in the entire debate?!? The issue is that his name has a space in it and we split the conversation by spaces thus "DE BLASIO" is never considered as we seek candidate inputs. Perhaps we should replace "DE BLASIO" with "DEBLASIO" to ensure his inputs are captured and we can still split by spaces, since everthing else seems to be working well. Let's handle this by adding a few lines of code to `analyzeCandidates` .

In [13]:
```
def switchDeBlasio(listOfWords):
    i = 0
    while i < len(listOfWords):
        if listOfWords[i] == "DE" and listOfWords[i + 1] == "BLASIO:":
            listOfWords.pop(i)
            listOfWords[i] = "DEBLASIO:"
        i += 1
    return listOfWords
```

In [14]:
```
def analyzeCandidates2(lst, night1=False):
    """
    input: List(str)= the conversations of each night, where each speaker has a co
lon in the last place
    output: dictionary mapping speakers to a list of their inputs during the nigh
t.
    """

    speakers = findSpeakers(lst, night1)
    speakersToText = {}
    for s in speakers:
        speakersToText[s] = []
        #print('\t', s)

    wordByWord = switchDeBlasio(''.join(lst).split())

    start = 0
    end = 1
    while end < len(wordByWord):
        speaker = wordByWord[start][:-1]
        mightSpeak = wordByWord[end]
        if isSpeaker(mightSpeak, speakers):
            currLst = speakersToText[speaker]
            currLst.append(' '.join(wordByWord[start + 1:end]))
            speakersToText[speaker] = currLst
            start = end
        end += 1
    return speakersToText
```

In [15]:
```
candidates1 = analyzeCandidates2(night1, True)
for candidate in candidates1.keys():
    print(candidate, "made", len(candidates1[candidate]), "contributions.")
```

```
TODD made 71 contributions.
GABBARD made 15 contributions.
GUTHRIE made 32 contributions.
(UNKNOWN) made 6 contributions.
DELANEY made 28 contributions.
HOLT made 40 contributions.
BOOKER made 18 contributions.
INSLEE made 9 contributions.
CASTRO made 23 contributions.
KLOBUCHAR made 15 contributions.
ANNOUNCER made 1 contributions.
MADDOW made 36 contributions.
RYAN made 21 contributions.
DIAZ-BALART made 28 contributions.
O’ROURKE made 28 contributions.
DEBLASIO made 13 contributions.
WARREN made 16 contributions.
```

Great, we've put that fire out! I'm going to remove the input that was attributed to the Announcer. Let's see what was said by unknown. I want to remove it, but it's important to ensure there aren't some juicy details there.

In [16]:
```
candidates1["(UNKNOWN)"]
```

Out[16]:
```
['That’s a false claim.',
 'Thank you.',
 'Jose…',
 'Yes, it was.',
 'Their mikes are on.',
 'Chuck…']
```

Looks pretty irrelevant to our analysis. So we'll ignore those inputs as well.

In [17]:
```
candidates1.pop('ANNOUNCER', None)
candidates1.pop('(UNKNOWN)', None)
```

Out[17]:
```
['That’s a false claim.',
 'Thank you.',
 'Jose…',
 'Yes, it was.',
 'Their mikes are on.',
 'Chuck…']
```

There were 5 moderators during the debates, so perhaps we should split this speakers set into moderators and candidates.

In [18]:
```
moderators = {'GUTHRIE', 'HOLT', 'MADDOW', 'TODD', 'DIAZ-BALART'}
candidates = set()
```

In [19]:
```
for speaker in list(candidates1):
    if speaker not in moderators:
        candidates.add(speaker)
len(candidates)
```

Out[19]:
```
10
```

Great! So now we have the candidates of the first evening all saved in the variable `candidates` . Now let's add the data from the second evening of debates!

In [20]:
```
candidates2 = analyzeCandidates2(night2)
for candidate in candidates2.keys():
    print(candidate, "made", len(candidates2[candidate]), "contributions.")
```

```
SANDERS made 49 contributions.
BIDEN made 43 contributions.
TODD made 88 contributions.
GUTHRIE made 46 contributions.
BENNET made 31 contributions.
HOLT made 45 contributions.
HICKENLOOPER made 10 contributions.
GILLIBRAND made 38 contributions.
UNKNOWN made 9 contributions.
WILLIAMSON made 26 contributions.
MADDOW made 43 contributions.
BALART made 16 contributions.
YANG made 7 contributions.
DIAZ-BALART made 30 contributions.
HARRIS made 45 contributions.
DIAZ BALART made 0 contributions.
SWALWELL made 33 contributions.
BUTTIGIEG made 22 contributions.
```

Seems like "DIAZ BALART" didnt make any contributions but he is a moderator, and his name is duplicated. So we'll add "DIAZ BALART" and "BALART" to our `moderators` set.

In [21]:
```
moderators.add("DIAZ BALART")
moderators.add("BALART")
```

Again, "UNKNOWN" made no import contributions, so we'll just drop this from our analysis.

In [22]:
```
candidates2.pop('UNKNOWN', None)
```

Out[22]:
```
['Yeah.',
 'Yeah.',
 '(INAUDIBLE) comment—',
 'been part of the issue—',
 '—Wait for evolution on these issues.',
 'Can I—',
 '(INAUDIBLE) lifetime, (INAUDIBLE) for all and today—',
 '—No—',
 '—That’s inventive enough—']
```

In [23]:
```
for speaker in list(candidates2):
    if speaker not in moderators:
        candidates.add(speaker)
len(candidates)
```

Out[23]:
```
20
```

Now, we have all of candidates in `candidates` and their inputs for each evening are stored in the variables `candidates1` and `candidates2` .

## Questions of Interest

1. What were the top 5 words used most frequently during the debate?
2. For each candidate, what were their 5 most frequently used words? How about top 5 bigrams? Top 5 trigrams?
3. How many words did each candidate say? Unique words? Average length of words used?
4. People invoked: How many times was Mitch McConnell mentioned? How about Trump? Nancy Pelosi? Barack Obama?
5. Locations invoked: How many times was North Korea mentioned? Russia? China? Iran? Sudan? Afghanistan?
6. How many times did a candidate mention another candidate, providing them at least 30 seconds to respond?
7. How positive was each candidate's message?

## 1: Top 5 Words

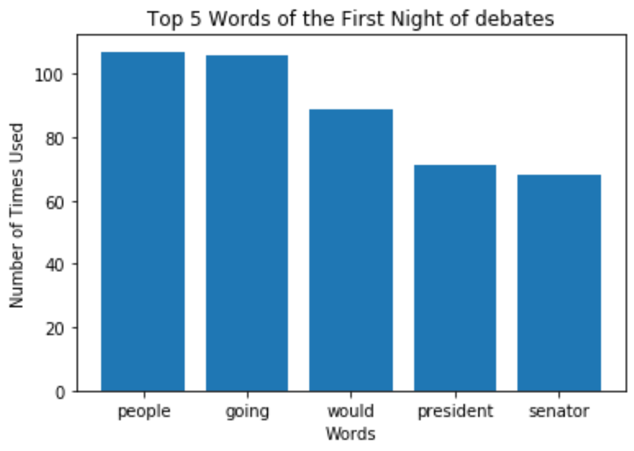
This analysis would be easier to conduct using `night1` and `night2` since they already contain all words from each evening. First we'll see what the top 5 words are from each night and we'll be able to if they discussed the same topics each night.

```
In [24]: apostrophe = nltk.word_tokenize(' '.join(night1))[8]
         apostrophe

Out[24]: '’'
```

In [25]:
```
def countWords(lstOfConvos):
    """
    Input: List(str) a list of conversations, not tokenized
    Output: dictionary (key: word, value: number of times word appeard in the conv
os)
    """
    stripped = removePunctuation(' '.join(lstOfConvos))
    tokens = removeContractions(nltk.word_tokenize(''.join(stripped)))
    stopWords = set(stopwords.words('english'))
    word_counts = defaultdict(int)
    for word in tokens:
        word = word.lower()
        if word not in stopWords and len(word) > 4 and word.upper() not in candida
tes \
            and word.upper() not in moderators:
            word_counts[word] += 1

    return word_counts

def topKWords(lstOfConvos, k=5):
    """
    Input: List(str) a list of conversations, not tokenized; k(int) number of word
s to return
    Output: tuple (top K words, number of times the top )
    """
    sorted_counts = sorted(countWords(lstOfConvos).items(), key=lambda kv: kv[1])

    words, counts = [], []
    for k, v in list(sorted_counts)[::-1][:k]:
        words.append(k)
        counts.append(v)

    return np.array(words), np.array(counts)

def findTopK(wordCounts):
    sorted_counts = sorted(wordCounts.items(), key=lambda kv: kv[1])
    topKWords = [k for k, _ in list(sorted_counts)[::-1][:k]]
    topKCounts = [v for _, v in list(sorted_counts)[::-1][:k]]

    return topKWords, np.array(topKCounts)

def removePunctuation(lstOfWords):
    table = str.maketrans('', '', string.punctuation)
    return [w.translate(table) for w in lstOfWords]

def removeContractions(tokens):
    i = 0
    while i < len(tokens):
        token = tokens[i]
        if token == apostrophe:
            tokens.pop(i + 1)
            tokens.pop(i)
            tokens.pop(i - 1)
        else:
            i += 1
    return tokens
```

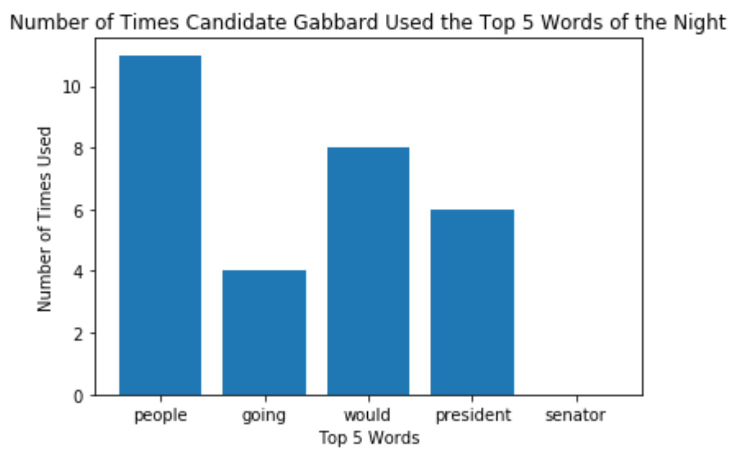
In [26]:
```
# maps candidates to a count of all the words they used
countsByCandidates = {}
for cand in candidates:
    try:
        countsByCandidates[cand] = countWords(candidates1[cand])
    except:
        countsByCandidates[cand] = countWords(candidates2[cand])
```

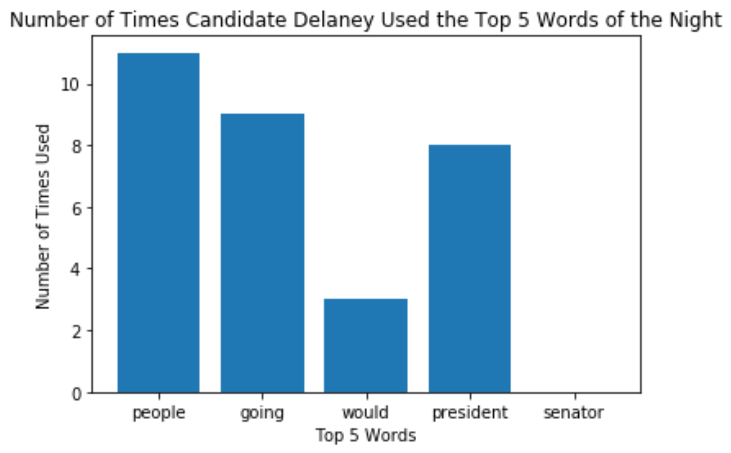
In [27]:
```
night1Top5, night1Top5counts = topKWords(night1)
plt.bar(night1Top5, night1Top5counts)
plt.title("Top 5 Words of the First Night of debates")
plt.xlabel("Words")
plt.ylabel("Number of Times Used");
```

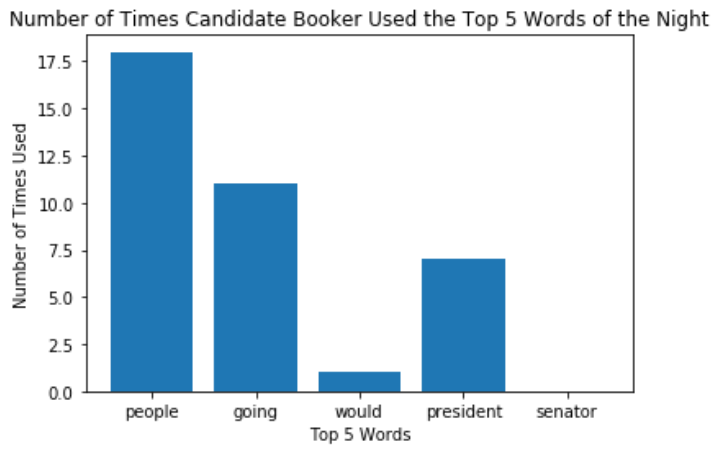


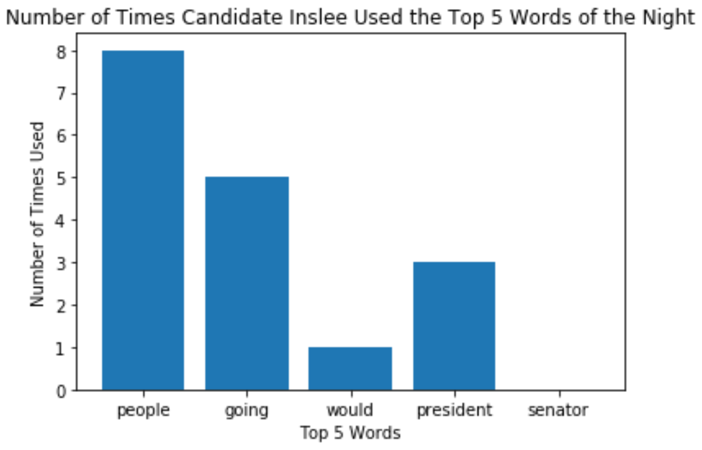
This analysis is including the words of the mooderators which may explain why 'president' and 'senator' are in the top 5 words as they present questions to the candidates. First, let's analyze the number of times each of the top 5 from each night appeared in each candidates. Then we'll see what the top 5 of each candidate was.

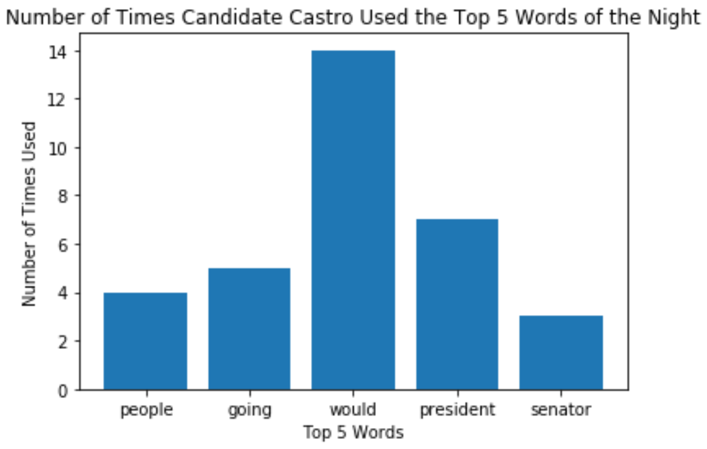
In [28]:
```
for speaker in candidates1.keys():
    if speaker in candidates:
        counts = countsByCandidates[speaker]
        candCount = {}
        for word in night1Top5:
            candCount[word] = counts[word]

        plt.bar(candCount.keys(), candCount.values())
        plt.title("Number of Times Candidate " + speaker.capitalize() + " Used the 
Top 5 Words of the Night")
        plt.xlabel("Top 5 Words")
        plt.ylabel("Number of Times Used")
    plt.show();
```

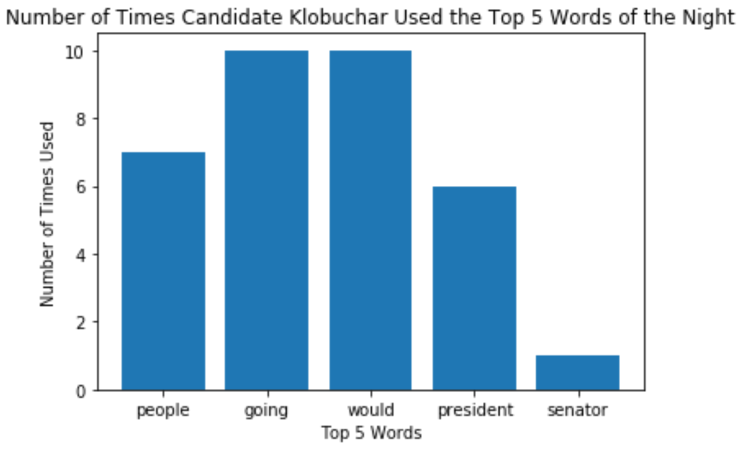

Number of Times Candidate Gabbard Used the Top 5 Words of the Night
Number of Times Used
0
2
4
6
8
10
people
going
would
president
senator
Top 5 Words

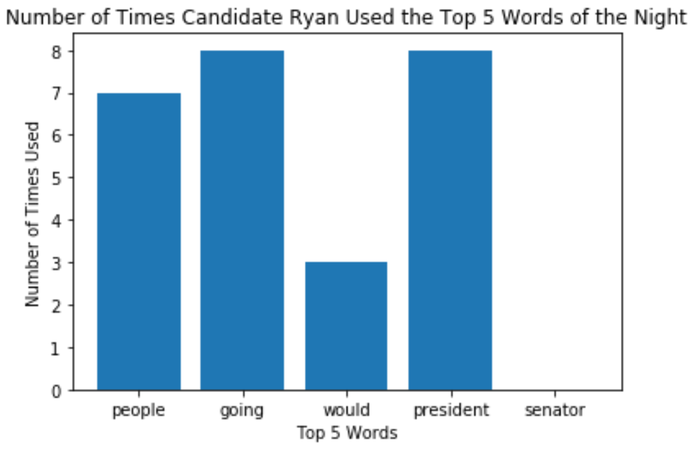

Number of Times Candidate Delaney Used the Top 5 Words of the Night
Number of Times Used
0
2
4
6
8
10
people
going
would
president
senator
Top 5 Words

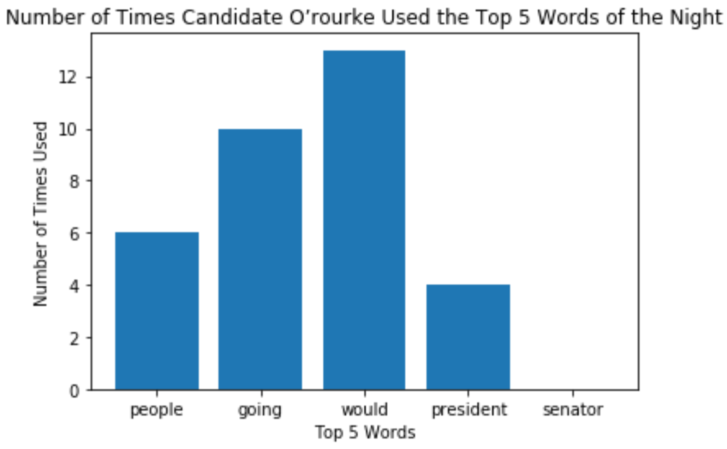

Number of Times Candidate Booker Used the Top 5 Words of the Night
Number of Times Used
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
people
going
would
president
senator
Top 5 Words

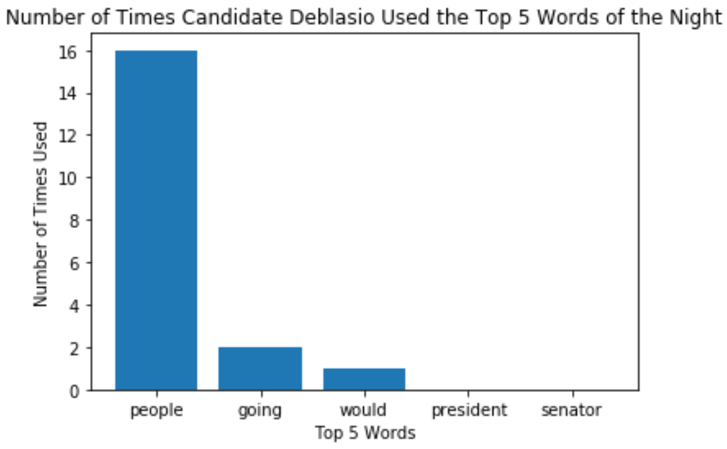

Number of Times Candidate Inslee Used the Top 5 Words of the Night
Number of Times Used
0
1
2
3
4
5
6
7
8
people
going
would
president
senator
Top 5 Words

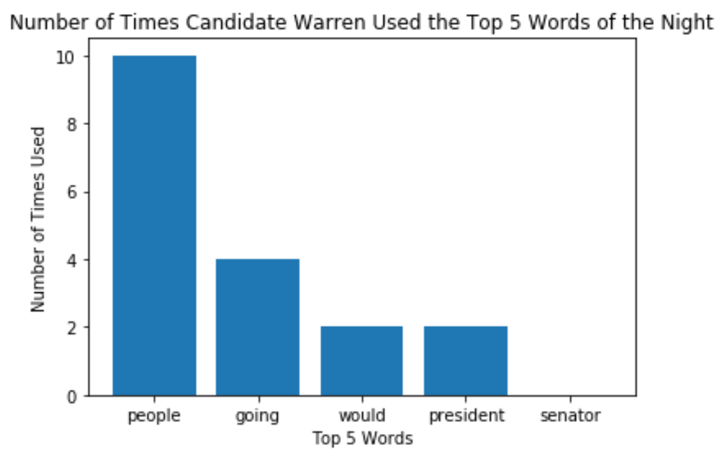

Number of Times Candidate Castro Used the Top 5 Words of the Night
Number of Times Used
0
2
4
6
8
10
12
14
people
going
would
president
senator
Top 5 Words


Number of Times Candidate Klobuchar Used the Top 5 Words of the Night
Number of Times Used
0
2
4
6
8
10
people
going
would
president
senator
Top 5 Words


Number of Times Candidate Ryan Used the Top 5 Words of the Night
Number of Times Used
Top 5 Words
people
going
would
president
senator
0
1
2
3
4
5
6
7
8


Number of Times Candidate O'rourke Used the Top 5 Words of the Night
Number of Times Used
Top 5 Words
people
going
would
president
senator
0
2
4
6
8
10
12

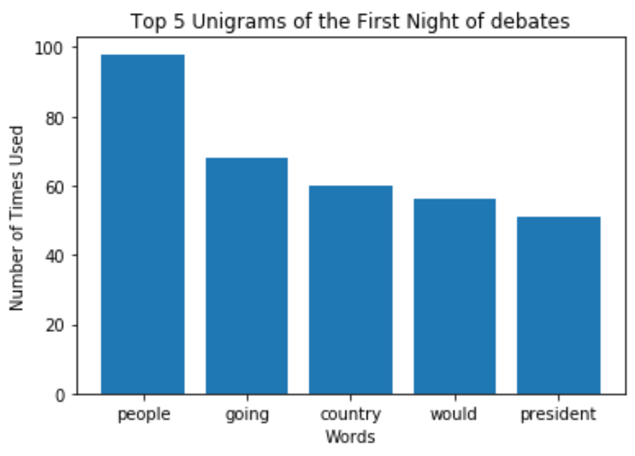

Number of Times Candidate Deblasio Used the Top 5 Words of the Night
Number of Times Used
Top 5 Words
people
going
would
president
senator
0
2
4
6
8
10
12
14
16



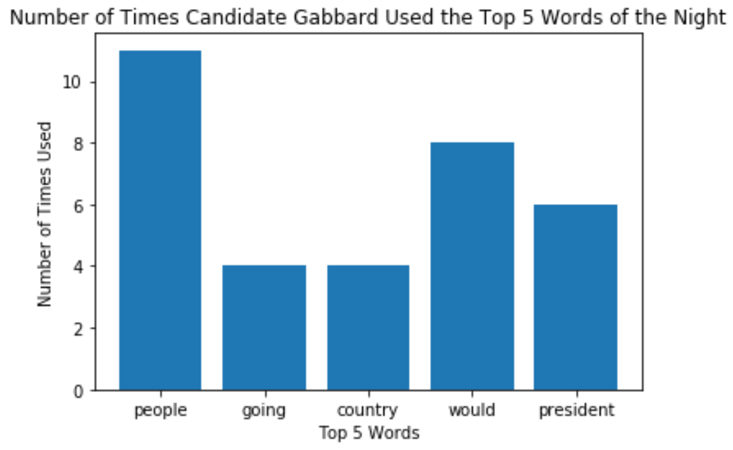
Only 2 candidates actually said senator, perhaps we should find the top 5 words of the debates and exclude words said by commentators.

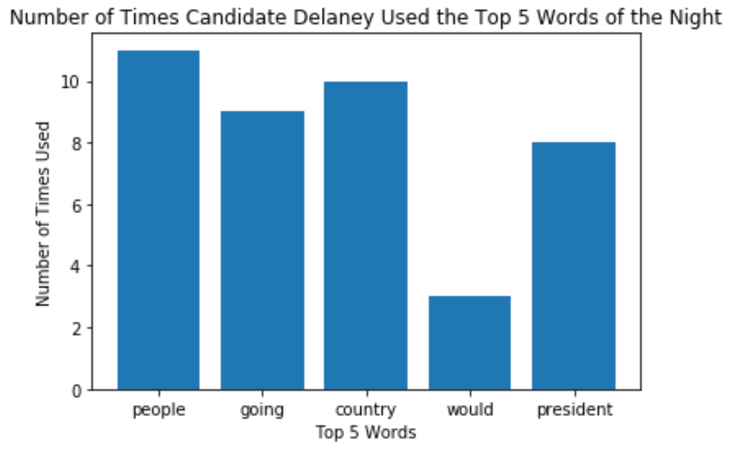
## 2. Top 5 Unigrams of each Candidate

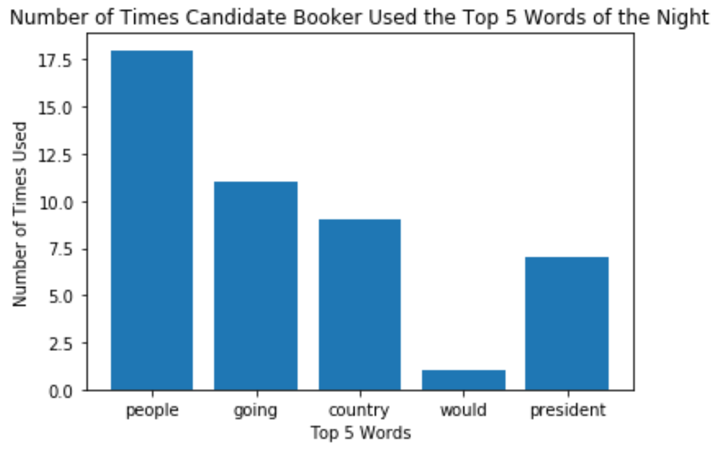
In [39]:
```
words = []
for cand in candidates:
    if cand in candidates1.keys():
        words += candidates1[cand]
candOnly, candOnlyCounts = topKWords(words)
plt.bar(candOnly, candOnlyCounts)
plt.title("Top 5 Unigrams of the First Night of debates")
plt.xlabel("Words")
plt.ylabel("Number of Times Used");
```

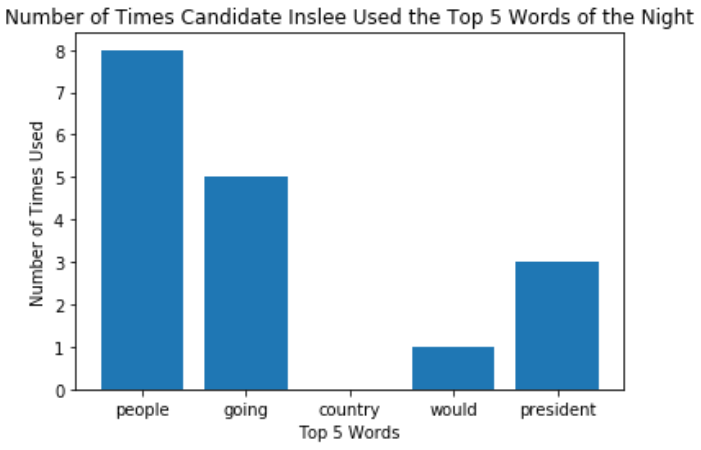


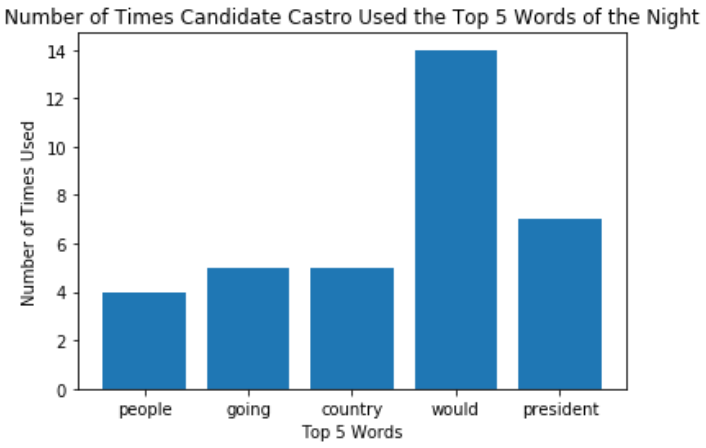
In [40]:
```
for speaker in candidates1.keys():
    if speaker in candidates:
        counts = countsByCandidates[speaker]
        candCount = {}
        for word in candOnly:
            candCount[word] = counts[word]

        plt.bar(candCount.keys(), candCount.values())
        plt.title("Number of Times Candidate " + speaker.capitalize() + " Used the
Top 5 Words of the Night")
        plt.xlabel("Top 5 Words")
        plt.ylabel("Number of Times Used")
    plt.show();
```

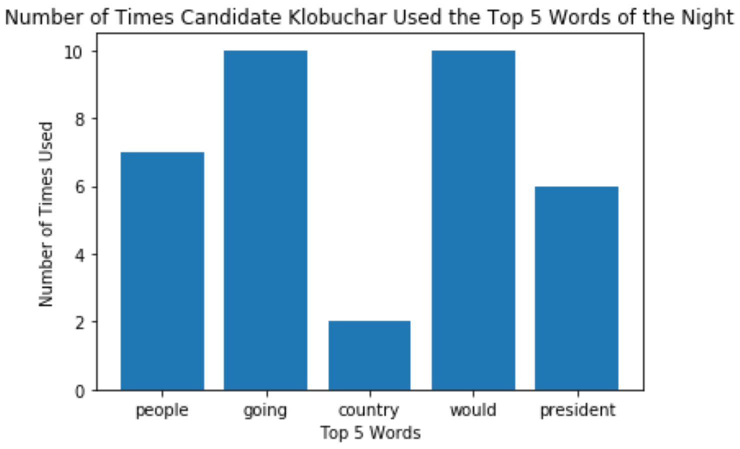

Number of Times Candidate Gabbard Used the Top 5 Words of the Night
Number of Times Used
0
2
4
6
8
10
people
going
country
would
president
Top 5 Words

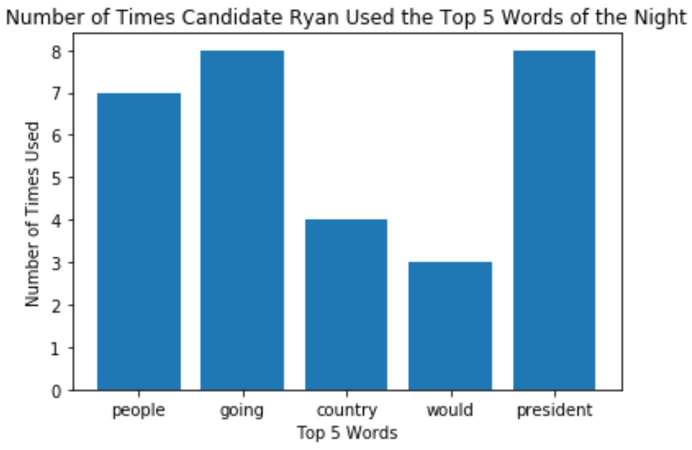

Number of Times Candidate Delaney Used the Top 5 Words of the Night
Number of Times Used
0
2
4
6
8
10
people
going
country
would
president
Top 5 Words

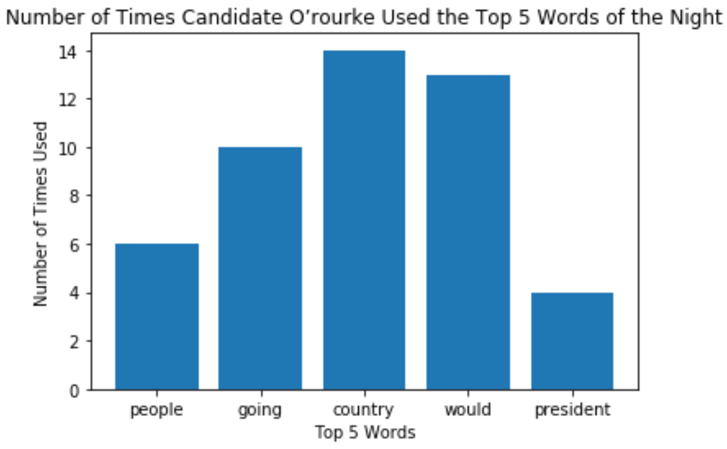

Number of Times Candidate Booker Used the Top 5 Words of the Night
Number of Times Used
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
people
going
country
would
president
Top 5 Words

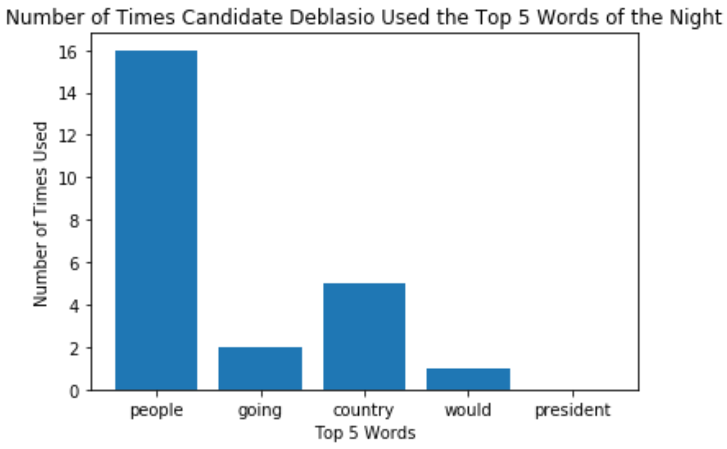

Number of Times Candidate Inslee Used the Top 5 Words of the Night
Number of Times Used
0
1
2
3
4
5
6
7
8
people
going
country
would
president
Top 5 Words

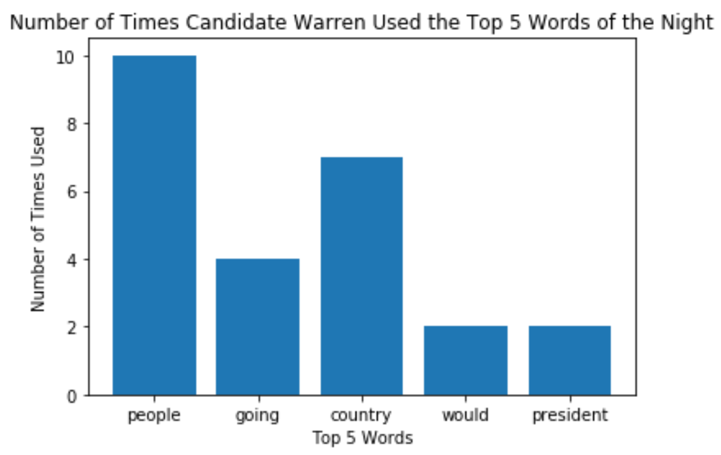

Number of Times Candidate Castro Used the Top 5 Words of the Night
Number of Times Used
0
2
4
6
8
10
12
14
people
going
country
would
president
Top 5 Words


Number of Times Candidate Klobuchar Used the Top 5 Words of the Night
Number of Times Used
0
2
4
6
8
10
people
going
country
would
president
Top 5 Words


Number of Times Candidate Ryan Used the Top 5 Words of the Night
Number of Times Used
0
1
2
3
4
5
6
7
8
people
going
country
would
president
Top 5 Words


Number of Times Candidate O'rourke Used the Top 5 Words of the Night
Number of Times Used
0
2
4
6
8
10
12
14
people
going
country
would
president
Top 5 Words

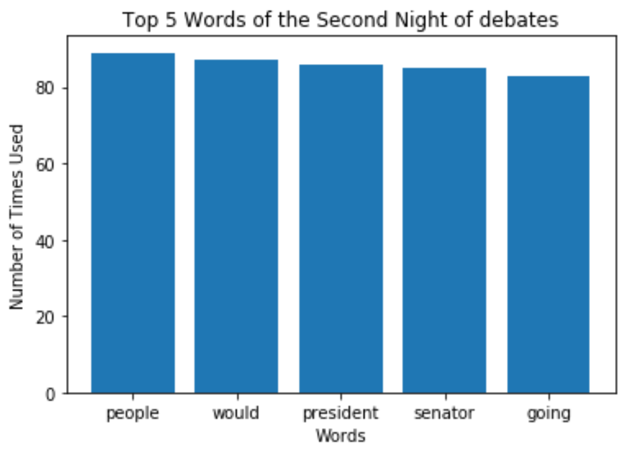

Number of Times Candidate Deblasio Used the Top 5 Words of the Night
Number of Times Used
0
2
4
6
8
10
12
14
16
people
going
country
would
president
Top 5 Words



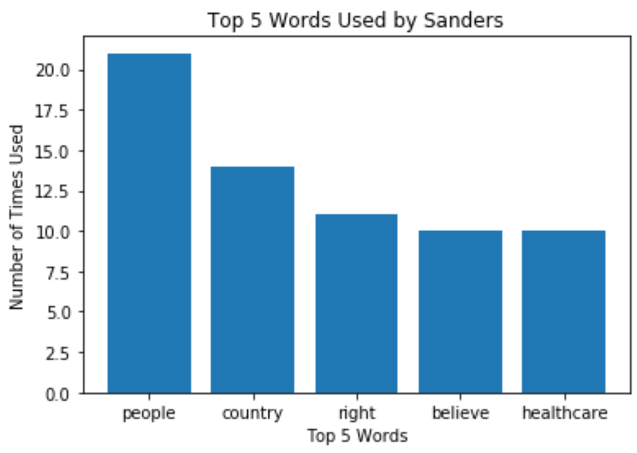
```
In [ ]:
```

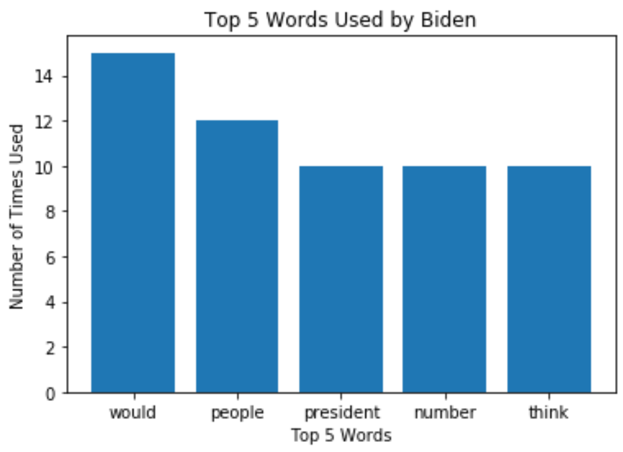
## Second Night

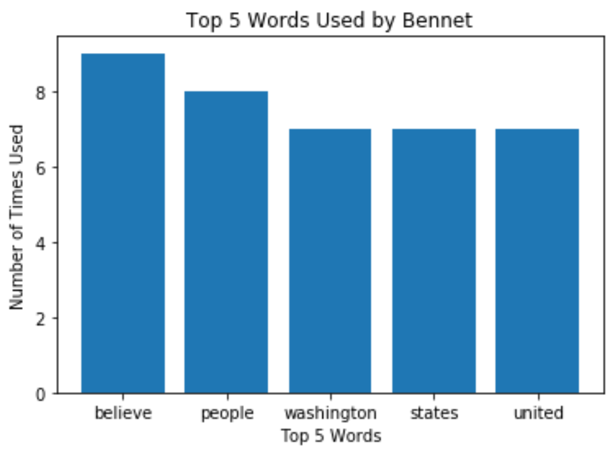
```
In [31]: night2Top5, night2Top5counts = topKWords(night2)
         plt.bar(night2Top5, night2Top5counts)
         plt.title("Top 5 Words of the Second Night of debates")
         plt.xlabel("Words")
         plt.ylabel("Number of Times Used");
```

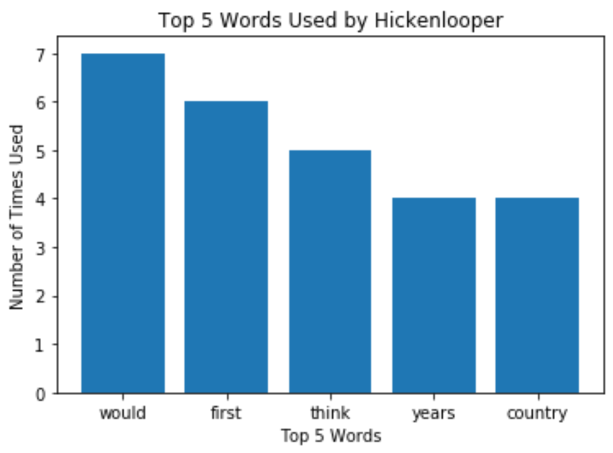


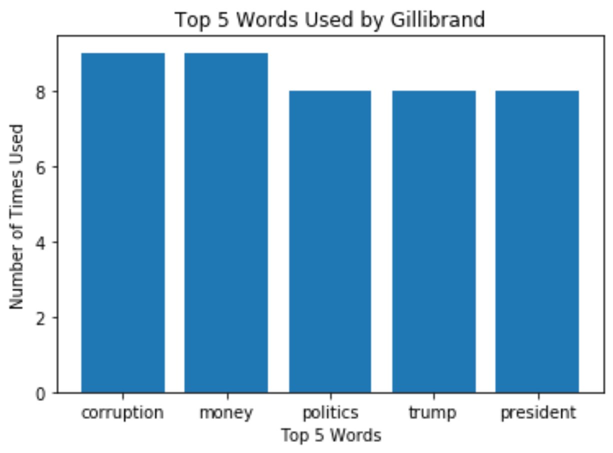
In [42]:
```
for candidate in candidates2.keys():
    if candidate in candidates:
        convos = candidates2[candidate]
        candidateTop5, candidateTop5counts = topKWords(convos)
        plt.bar(candidateTop5, candidateTop5counts)
        plt.title("Top 5 Words Used by " + candidate.capitalize())
        plt.xlabel("Top 5 Words")
        plt.ylabel("Number of Times Used")
    plt.show();
```

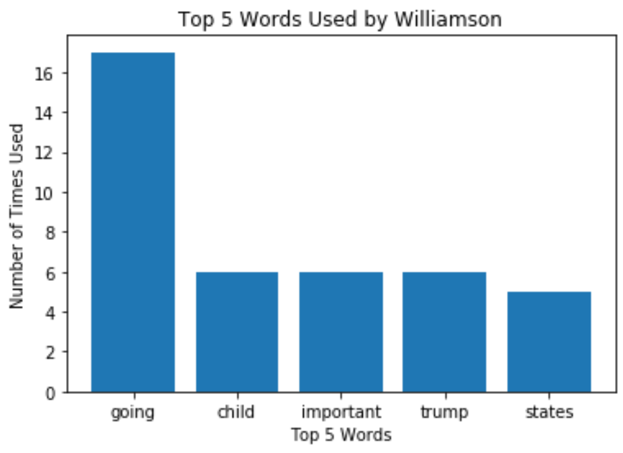

Top 5 Words Used by Sanders
Number of Times Used
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
people
country
right
believe
healthcare
Top 5 Words

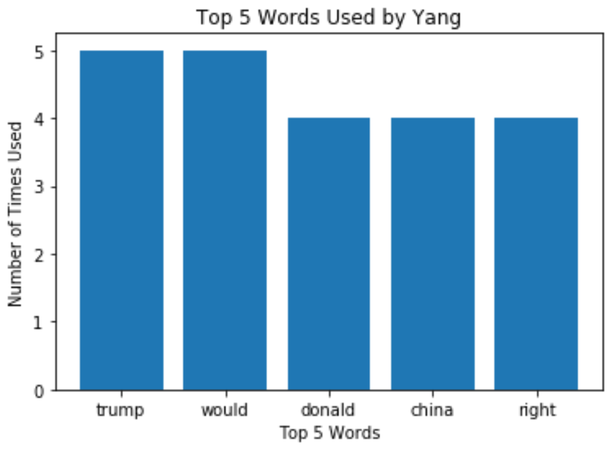

Top 5 Words Used by Biden
Number of Times Used
0
2
4
6
8
10
12
14
would
people
president
number
think
Top 5 Words

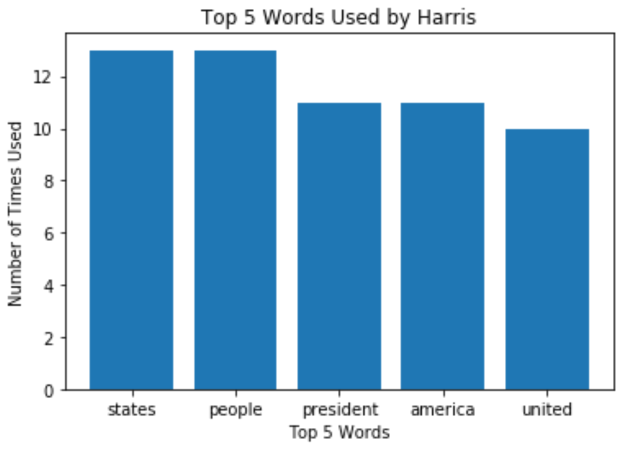

Top 5 Words Used by Bennet
Number of Times Used
0
2
4
6
8
believe
people
washington
states
united
Top 5 Words

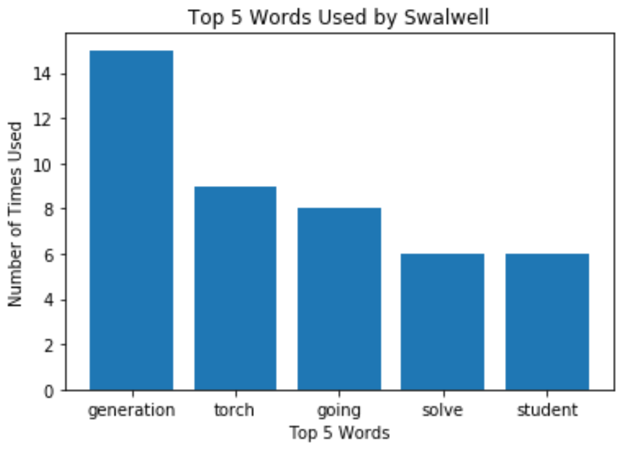

Top 5 Words Used by Hickenlooper
Number of Times Used
0
1
2
3
4
5
6
7
would
first
think
years
country
Top 5 Words

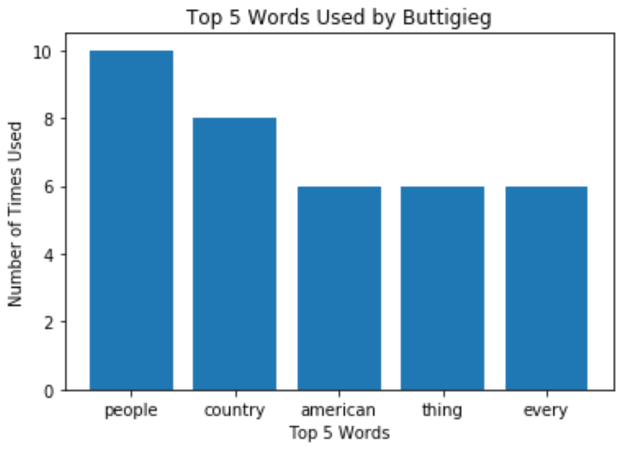

Top 5 Words Used by Gillibrand
Number of Times Used
0
2
4
6
8
corruption
money
politics
trump
president
Top 5 Words


Top 5 Words Used by Williamson
Number of Times Used
0
2
4
6
8
10
12
14
16
going
child
important
trump
states
Top 5 Words

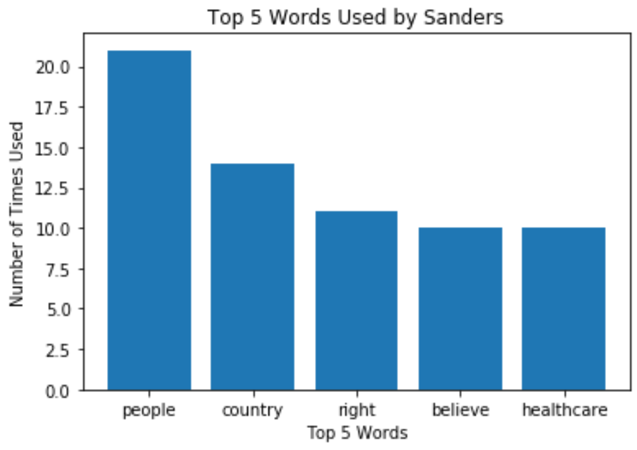

Top 5 Words Used by Yang
Number of Times Used
0
1
2
3
4
5
trump
would
donald
china
right
Top 5 Words

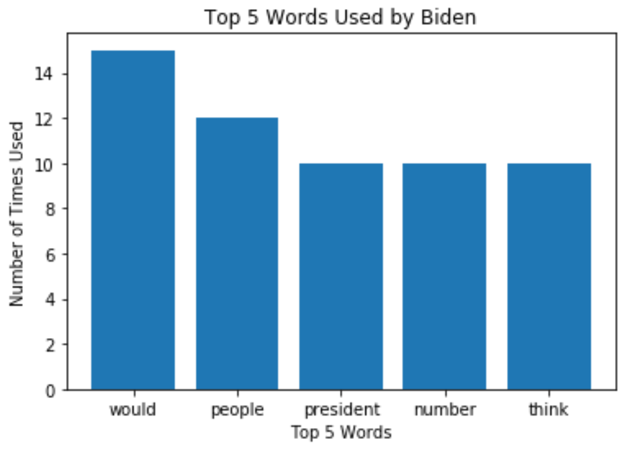

Top 5 Words Used by Harris
Number of Times Used
0
2
4
6
8
10
12
states
people
president
america
united
Top 5 Words

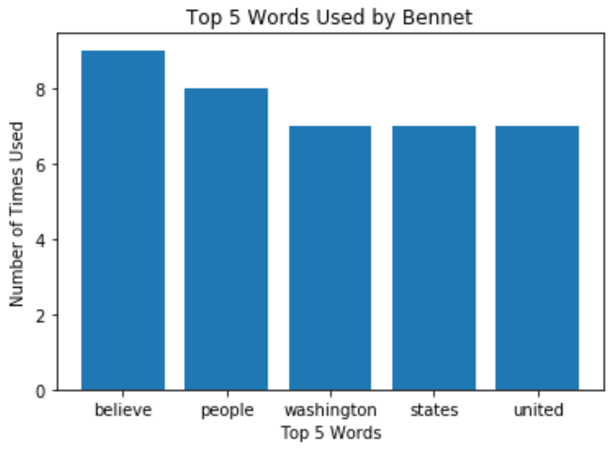

Top 5 Words Used by Swalwell
Number of Times Used
0
2
4
6
8
10
12
14
generation
torch
going
solve
student
Top 5 Words

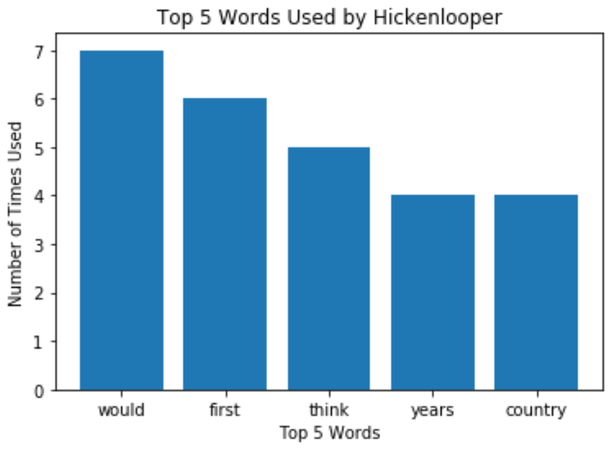

Top 5 Words Used by Buttigieg
Number of Times Used
Top 5 Words
people
country
american
thing
every
0
2
4
6
8
10

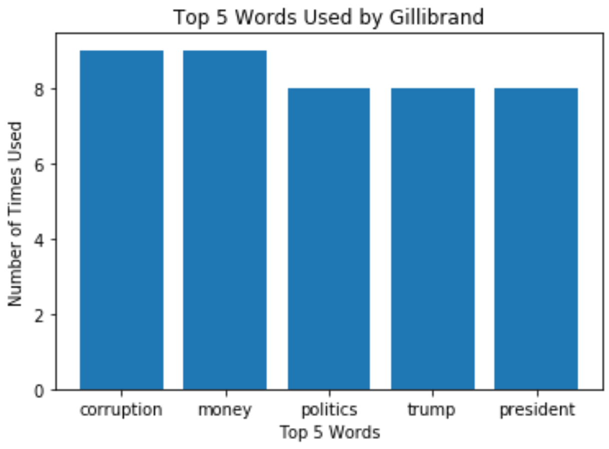
In [33]:
```
for candidate in candidates2.keys():
    if candidate in candidates:
        convos = candidates2[candidate]
        candidateTop5, candidateTop5counts = topKWords(convos)
        plt.bar(candidateTop5, candidateTop5counts)
        plt.title("Top 5 Words Used by " + candidate.capitalize())
        plt.xlabel("Top 5 Words")
        plt.ylabel("Number of Times Used")
    plt.show();
```

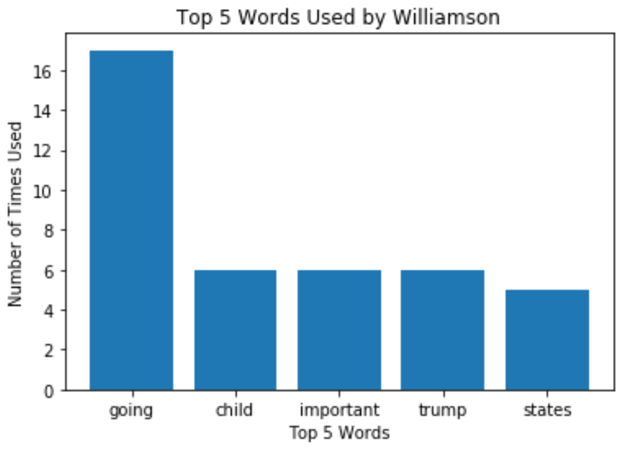

Top 5 Words Used by Sanders
Number of Times Used
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
people
country
right
believe
healthcare
Top 5 Words

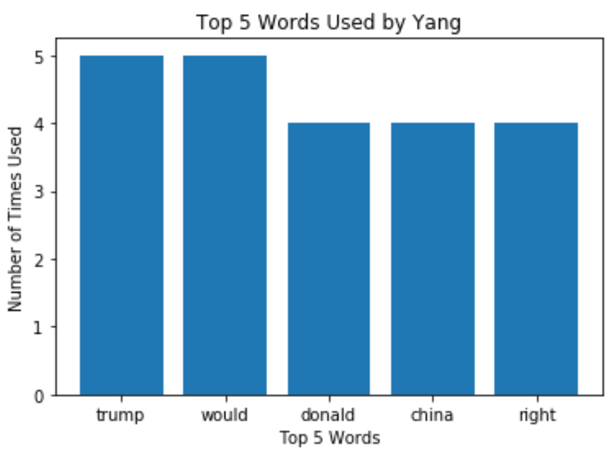

Top 5 Words Used by Biden
Number of Times Used
0
2
4
6
8
10
12
14
would
people
president
number
think
Top 5 Words

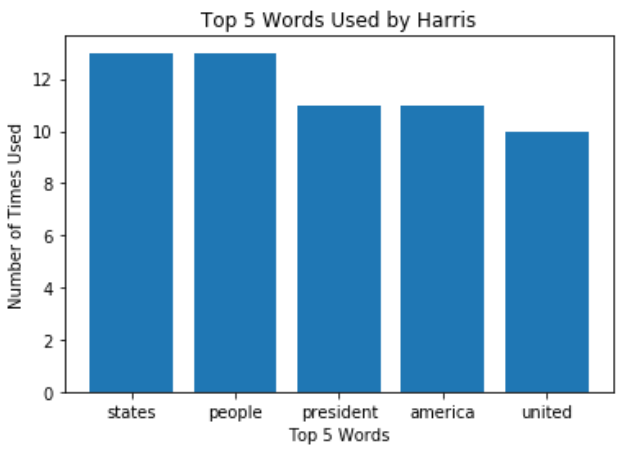

Top 5 Words Used by Bennet
Number of Times Used
0
2
4
6
8
believe
people
washington
states
united
Top 5 Words

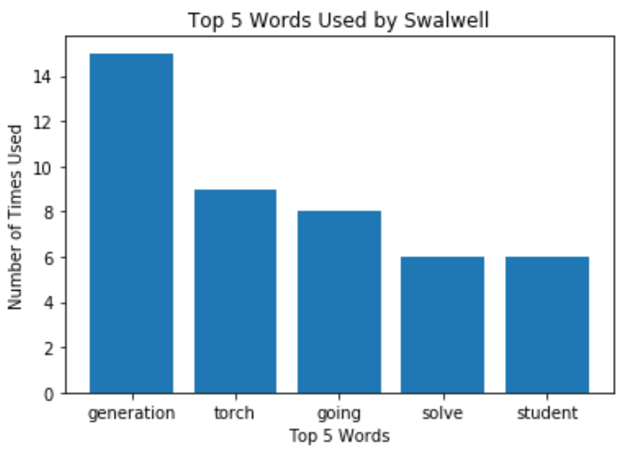

Top 5 Words Used by Hickenlooper
Number of Times Used
0
1
2
3
4
5
6
7
would
first
think
years
country
Top 5 Words

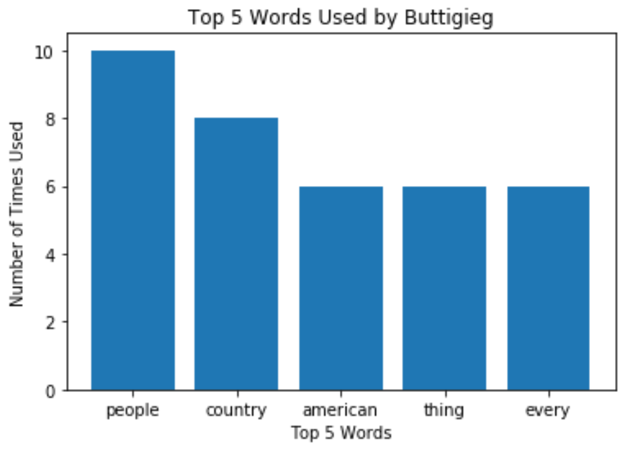

Top 5 Words Used by Gillibrand
Number of Times Used
0
2
4
6
8
corruption
money
politics
trump
president
Top 5 Words


Top 5 Words Used by Williamson
Number of Times Used
0
2
4
6
8
10
12
14
16
going
child
important
trump
states
Top 5 Words


Top 5 Words Used by Yang
Number of Times Used
0
1
2
3
4
5
trump
would
donald
china
right
Top 5 Words


Top 5 Words Used by Harris
Number of Times Used
0
2
4
6
8
10
12
states
people
president
america
united
Top 5 Words


Top 5 Words Used by Swalwell
Number of Times Used
0
2
4
6
8
10
12
14
generation
torch
going
solve
student
Top 5 Words



## 3. Word Counts

```
In [ ]:
```

```
In [ ]:
```

## 7 : Sentiment Analysis

It turns out that we can use the words in each candidate's response to calculate a measure of their sentiment of the statement. For example, the sentence "I love America!" has positive sentiment, whereas the sentence "I hate taxes!" has a negative sentiment. In addition, some words have stronger positive / negative sentiment than others: "I love America." is more positive than "I like America."

We will use the VADER (Valence Aware Dictionary and sEntiment Reasoner) (https://github.com/cjhutto/vaderSentiment) lexicon to analyze the sentiment of their statements. VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media which is great for our usage.

The VADER lexicon gives the sentiment of individual words. Run the following cell to show the first few rows of the lexicon:

```
In [34]: print(''.join(open("vader_lexicon.txt").readlines()[:10]))
```

```
$:      -1.5    0.80623 [-1, -1, -1, -1, -3, -1, -3, -1, -2, -1]
%)      -0.4    1.0198  [-1, 0, -1, 0, 0, -2, -1, 2, -1, 0]
%-)     -1.5    1.43178 [-2, 0, -2, -2, -1, 2, -2, -3, -2, -3]
&-:     -0.4    1.42829 [-3, -1, 0, 0, -1, -1, -1, 2, -1, 2]
&:      -0.7    0.64031 [0, -1, -1, -1, 1, -1, -1, -1, -1, -1]
( '}{' )        1.6     0.66332 [1, 2, 2, 1, 1, 2, 2, 1, 3, 1]
(%      -0.9    0.9434  [0, 0, 1, -1, -1, -1, -2, -2, -1, -2]
('-:    2.2     1.16619 [4, 1, 4, 3, 1, 2, 3, 1, 2, 1]
(':     2.3     0.9     [1, 3, 3, 2, 2, 4, 2, 3, 1, 2]
((-:    2.1     0.53852 [2, 2, 2, 1, 2, 3, 2, 2, 3, 2]
```

As you can see, the lexicon contains emojis too! The first column of the lexicon is the token, or the word itself. The second column is the polarity of the word, or how positive / negative it is.

(How did they decide the polarities of these words? What are the other two columns in the lexicon? See the link above.)

Next, we'll read the lexicon into a DataFrame called sent. The indices of the DF are the tokens in the lexicon.

In [35]:
```
tokens, polarites = [], []
for line in open("vader_lexicon.txt").readlines():
    line = line.split("\t")
    token, polarity = line[0], float(line[1])
    tokens.append(token)
    polarites.append(polarity)
sent = pd.DataFrame()
sent['token'] = tokens
sent['polarity'] = polarites
sent = sent.set_index('token')
sent.tail(25)
```

Out[35]:

| token | polarity |
|---|---|
| yeees | 1.7 |
| yep | 1.2 |
| yes | 1.7 |
| youthful | 1.3 |
| yucky | -1.8 |
| yummy | 2.4 |
| zealot | -1.9 |
| zealots | -0.8 |
| zealous | 0.5 |
| {: | 1.8 |
| \|-0 | -1.2 |
| \|-: | -0.8 |
| \|-:> | -1.6 |
| \|-o | -1.2 |
| \|: | -0.5 |
| \|;-) | 2.2 |
| \|= | -0.4 |
| \|^: | -1.1 |
| \|o: | -0.9 |
| \|\|-: | -2.3 |
| }: | -2.1 |
| }:( | -2.0 |
| }:) | 0.4 |
| }:-( | -2.1 |
| }:-) | 0.3 |

```
In [36]: sent.loc["disagree"]["polarity"]

Out[36]: -1.6
```

In the lexicon, all the tokens are lowercase so lets change all of the candidates speech to lowercase. Below I've made some helper methods to assist with our Sentiment Analysis. Since lexicon has no use for punctuation, we must remove them.

```
In [37]: tokens = set(sent.reset_index()["token"])
         def removePunctuations(s):
             pattern = re.compile(r'[^\w\s]')
             return pattern.sub('', s)

         def findPolarity(lst_of_words):
             ret = 0
             for word in lst_of_words:
                 if word in tokens:
                     ret += sent.loc[word]["polarity"]
             return ret
```

```
In [38]: polarities = {}
         for speaker in speakers:
             for speech in speakersToText2[speaker]:
                 if speaker in polarities:
                     lst = polarities[speaker]
                 else:
                     lst = []
                 lst.append(findPolarity(removePunctuations(speech.lower()).split()))
                 polarities[speaker] = lst

         ---------------------------------------------------------------------------
         NameError                                 Traceback (most recent call last)
         <ipython-input-38-e1be5c1f733e> in <module>()
               1 polarities = {}
         ----> 2 for speaker in speakers:
               3     for speech in speakersToText2[speaker]:
               4         if speaker in polarities:
               5             lst = polarities[speaker]

         NameError: name 'speakers' is not defined
```

```
In [ ]: for speaker in speakers:
            print(speaker, np.mean(polarities[speaker]))
```

```
In [ ]: findPolarity(removePunctuations(speakersToText2["BOOKER"][0]).lower().split())
```

```
In [ ]:
```